SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Juraj Šikonja

RJEŠAVANJE KOMBINATORNIH PROBLEMA POMOĆU TOKOVA U MREŽAMA

Diplomski rad

Voditelj rada:
Prof.dr.sc. Robert Manger

Zagreb, 2017

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Posvečujem ovaj rad mojoj djevojci Kristini i mojim roditeljima Željku i Marini. Kristini, koja je bila sa mnom i u dobru i u zlu sve ove godine. Bila je tu kada mi je bilo teško i dizala me kada sam padao. Roditeljima koji su me cijeli život usmjeravali i poticali, uvjeravali me da mogu više i da uvijek budem bolji i kojima dugujem sve što sam danas i što ću jednog dana biti.

Sadržaj

Uvod

Mreže i mrežni tokovi su zanimljivo područje matematike iz kojeg proizlaze mnoge primjene na svakodnevne probleme iz područja: računarstva, ekonomije, poslovnog svijeta, ali i mnogih drugih prodručja. U ovom radu ćemo istražiti područje primjene mreža i mrežnih tokova za rješavanje odabranih kombinatornih problema.

Najprije ćemo postaviti osnove definiranjem osnovnih pojmova vezanih za neusmjerene i usmjerene grafove, mreže i mrežne tokove, da bi nakon toga predstavili centralni problem mrežnih tokova, a to je pronalaženje maksimalnog toka u danoj mreži. Nakon što predstavimo dva algoritma koja se koriste za rješavanje tog problema, predstaviti ćemo nekoliko kombinatornih problema koji se mogu efikasno rješavati pomoću tokova u mrežama.

Na kraju ćemo na dva praktična primjera pokazati kako primjeniti tokove u mrežama za njihovo rješavanje, uspoređujući pritom efikasnost dva prethodno predstavljena algoritma.

Poglavlje 1

Osnovni pojmovi i definicije

Prije nego što započnemo govoriti o mrežama i tokovima u mrežama te prije bilo kakvog govora o primjenama istih moramo uvesti neke osnovne pojmove vezane uz grafove i mreže. Najprije ćemo definirati neusmjereni graf i pojmove vezane za neusmjerene grafove, potom ćemo isto učiniti za usmjerene grafove, što će nam omogućiti da uvedemo pojmove mreža i tokova u mrežama.

1.1 Neusmjereni i usmjereni grafovi

Definicije u sljedećem poglavlju su preuzete iz [12].

Definicija 1.1.1. *Neka je* $V = \{v_i | i = 1, 2, \ldots, n\}$ *proizvoljan konačan skup. Neka je nadalje* S *skup svih neuređenih parova različitih elemenata iz* V*, tj.*

$$S = \{\{v_i, v_j\} | v_i \in V, v_j \in V, v_i \neq v_j\},$$

pri ćemu $\{v_i, v_j\}$ *i* $\{v_j, v_i\}$ *predstavljaju jedan te isti element.*

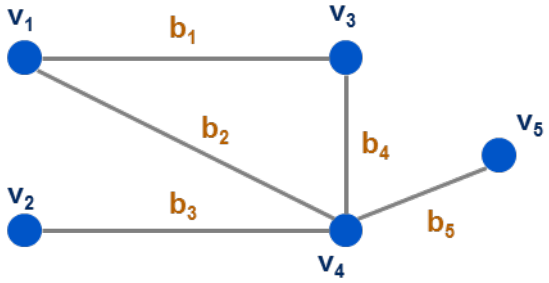
Uređeni par $G = (V, B)$*, gdje je* $B \subseteq S$*, nazivamo* ***neusmjerenim grafom****. Pritom elemente iz* V *nazivamo čvorovi, dok elemente iz* B *nazivamo bridovi grafa.*

Primjer 1.1.2. *Neka je* $V = \{v_1, v_2, v_3, v_4, v_5\}$ *i* $B = \{b_1, b_2, b_3, b_4, b_5\} = \{\{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_4, v_5\}\}$*. Neusmjereni graf* $G = (V, B)$ *je prikazan na Slici 1.1*

Za dva čvora kažemo da su **susjedna** ako postoji brid kojem su oni krajnje točke.

Stupanj čvora u neusmjerenom grafu jednak je broju bridova kojima je taj čvor jedan od krajnjih točaka.

Primjer 1.1.3. *Na grafu na Slici 1.1 čvor* v_3 *ima stupanj* 2*, jer je krajnja točka bridovima* b_1 *i* b_4*.*



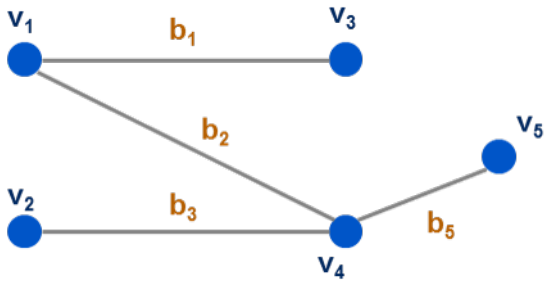
Slika 1.1: Neusmjereni graf

Stupanj neusmjerenog grafa jednak je maksimalnom stupnju među stupnjevima svih čvorova grafa. Prethodni graf ima stupanj 4 jer je čvor v_4 stupnja 4, a to je ujedno maksimalan stupanj svih čvorova grafa.

Šetnja iz čvora v_1 u čvor v_s u neusmjerenom grafu $G = (V, B)$ je konačan niz čvorova $v_1, v_2, \ldots, v_s$ i bridova takvih da je $\{v_k, v_{k+1}\} \in B$, $k = 1, 2, \ldots, s-1$. Pritom se čvor v_1 naziva **početni čvor** (ili izvor) šetnje, dok se v_s naziva **završni čvor** (ili ponor) šetnje. Broj čvorova u nizu koji definira šetnju nazivamo duljinom šetnje, a ona je jednaka broju čvorova minus 1. Šetnja je zatvorena, ako je početni čvor jednak završnom čvoru.

Primjer 1.1.4. *Šetnja definirana sa nizom čvorova i bridova* $v_5, b_5, v_4, b_2, v_1, b_1, v_3$ *neusmjerenom grafu sa Slike 1.1 je šetnja iz čvora* v_5 *u čvor* v_3 *duljine 3.*

Staza je šetnja u kojoj se bridovi ne ponavljaju. **Put** je šetnja u kojoj se čvorovi ne ponavljaju. **Ciklus** je šetnja $v_1, v_2, \ldots, v_s$ u kojoj su čvorovi $v_1, v_2, \ldots v_{s-1}$ različiti (pa čine put), te vrijedi v_s=v_1.



Slika 1.2: Razapinjuće stablo

Primjer 1.1.5. *Niz čvorova i bridova $v_4, b_4, v_3, b_1, v_1, b_2, v_4$ na neusmjerenom grafu sa Slike 1.1 je ciklus.*

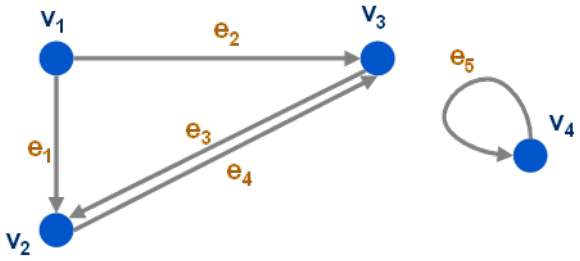
Podgraf grafa $G = (V, B)$ je graf s čvorovima koji pripadaju skupu čvorova V, a bridovi pripadaju skupu bridova B.

Neusmjeren graf je **povezan** ako za svaki par različitih čvorova v_i, $v_j \in V$ grafa postoji put iz v_i u v_j. Povezan neusmjeren graf nazivamo **stablo** ako ne sadržava niti jedan ciklus.

Podgraf grafa $G = (V, B)$ koji je stablo, a skup čvorova mu je V, nazivamo **razapinjućim stablom grafa** G.

Primjer 1.1.6. *Stablo na Slici 1.2 definirano sa $V = \{v_1, v_2, v_3, v_4, v_5\}$ i $B = \{b_1, b_2, b_3, b_5\} = \{\{v_1, v_3\}, \{v_1, v_4\}, \{v_2, v_4\}, \{v_4, v_5\}\}$ je razapinjuće stablo grafa na Slici 1.1*

Definicija 1.1.7. *Neka je $V = \{v_i | i = 1, 2, 3, \ldots, n\}$ proizvoljan konačan skup te neka je $V \times V = \{(v_i, v_j) | v_i \in V, v_j \in V\}$ skup svih uređenih parova elemenata iz V. Uređeni par $G = (V, E)$, gdje je $E \subseteq V \times V$, nazivamo **usmjerenim grafom**. Elementi od V su čvorovi, dok su elementi iz E bridovi usmjerenog grafa G.*



Slika 1.3: Usmjereni graf

Primjer 1.1.8. *Neka je $V = \{v_1, v_2, v_3, v_4\}$ i $E = \{e_1, e_2, e_3, e_4, e_5\} = \{(v_1, v_2), (v_1, v_3), (v_3, v_2), (v_2, v_3), (v_4, v$
Usmjereni graf $G = (V, E)$ je prikazan na Slici 1.3.*

Kažemo da su čvorovi v_i i v_j krajnje točke brida $e = (v_i, v_j)$, pri čemu je v_i početna, a v_j završna točka brida. Kažemo da brid e izlazi iz čvora v_i, ulazi u čvor v_j te da je incidentan s čvorovima v_i i v_j. U usmjerenom grafu $G = (V, E)$ za dva brida kažemo da su susjedna ako su incidentna s istim čvorom.

Stupanj $d(i)$ nekog čvora v u usmjerenom grafu jednak je broju bridova incidentnih s tim čvorom. Broj bridova kojima je čvor v početna točka označimo sa $d^+(v)$, a broj bridova kojima je čvor v završna točka označimo sa $d^-(v)$.

Ako je $d(v) = 0$, kažemo da je v **izolirani čvor**. Čvor v za koji vrijedi $d(v) = 1$ nazivamo ekstremnim čvorom grafa. Ako je $d^+(v) \neq 0$ i $d^-(v) = 0$, tada je čvor v izvor (ulaz) grafa. Ako je $d^+(v) = 0$ i $d^-(v) \neq 0$, čvor v je ponor (izlaz) grafa. Graf može imati više izvora i ponora.

Primjer 1.1.9. *U usmjerenom grafu prikazanom na Slici 1.3* $d(v_2) = 3, d^+(v_2) = 1$, *te* $d^-(v_2) = 2$. *Izvor je* v_1, *dok graf nema ponora.*

Šetnja u usmjerenom grafu $G = (V, E)$ je niz čvorova $v_1, v_2, \ldots, v_s$, s pripadnim nizom bridova $e_1, e_2, \ldots, e_{s-1}$, takvih da je za $k = 1, 2, \ldots, s-1$ ili $e_k = (v_k, v_{k+1})$ (brid usmjeren prema naprijed) ili $b_k = (v_{k+1}, v_k)$ (brid usmjeren prema nazad). Napomenimo, ako su čvorovi v_k i v_{k+1} jedan za drugim u šetnji te ako su oba brida (v_k, v_{k+1}) i (v_{k+1}, v_k) u promatranom grafu, tada se bilo koji od njih moze upotrijebiti u šetnji.

Put je šetnja u kojoj su svi čvorovi $v_1, v_2, \ldots, v_s$ različiti.

Ciklus je šetnja s različitim čvorovima $v_1, v_2, \ldots, v_s$ u kojoj je početni jednak završnom čvoru, tj. $v_1 = v_s$.

Kažemo da je šetnja u usmjerenom grafu $G = (V, E)$ usmjerena ako sadržava samo naprijed usmjerene bridove. Također kažemo da je put, odnosno ciklus, usmjeren ako sadržava samo naprijed orijentirane bridove.

Petlja je ciklus koji se sastoji od samo jednog brida (v_i, v_i). Na grafu na slici 1.3 to je brid e_5.

Duljina puta je broj bridova što ga čine. Petlja je kružni put duljine 1.

Polazeći od usmjerenog grafa $G = (V, E)$ može se konstruirati odgovarajući neusmjereni graf tako da se zanemere smjerovi na bridovima i umjesto dva brida, koji povrezuju neka dva čvora, stavi se jedan brid. Kažemo da je usmjereni graf povezan ako je odgovarajući neusmjeren graf povezan.

1.2 Mreže i tokovi u mrežama

Definicija 1.2.1. *Mrežu N definiramo kao uređenu četvorku* (G, s, t, c) *gdje je:*

- G *konačni usmjereni graf* (V, E) *koji ne sadrži petlje i paralelne bridove.*
- s *i* t *dva određena čvora, gdje se* s *nazivamo izvor a* t *ponor.*
- c *funkcija kapaciteta,* $c : E \to \mathbb{R}^+$. *Pozitivan realan broj* $c(e)$, *se naziva kapacitet brida* e.

Za svaki čvor $v \in V$ *neka* $\alpha(v)$ *označava skup bridova koji ulaze u* v *iz* G. *Slično, neka* $\beta(v)$ *označava skup bridova koji izlaze iz* v *u* G.

Funkcija toka $f : E \to \mathbb{R}$, je dodjeljivanje realnih brojeva $f(e)$ svakom bridu e tako da su zadovoljena sljedeća dva pravila:

***Pravilo brida**: Za svaki brid $e \in E, 0 \leq f(e) \leq c(e)$.*

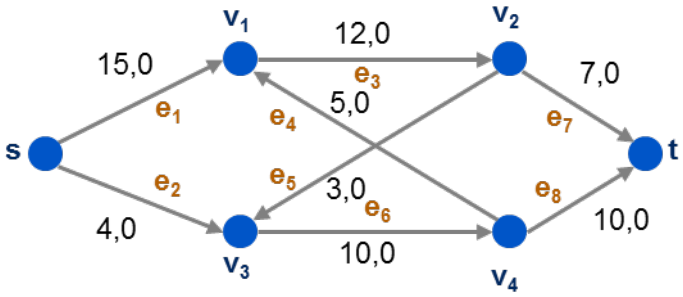
***Pravilo čvora**: Za svaki čvor $v \in V \setminus \{s, t\}$,*

$$\sum_{e \in \alpha(v)} f(e) = \sum_{e \in \beta(v)} f(e).$$

Ukupni tok F, od f u N, je definiran sa

$$F = \sum_{e \in \alpha(t)} f(e) - \sum_{e \in \beta(t)} f(e). \tag{1.1}$$

Naime, F je ukupni zbroj svih tokova u ponor.



Slika 1.4: Mreža

Primjer 1.2.2. *Neka je $V = \{s, v_1, v_2, v_3, v_4, t\}$ i $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\} = \{(s, v_1), (s, v_3), (v_1, v_2), (v_4, v_1), (v_2, v_3), (v_3, v_4), (v_2, t), (v_4, t)\}$.*
Neka je G(V,E) usmjereni graf. Neka je c funkcija $c : E \to \mathbb{R}^+$ definirana sa $c(e_1) = 15$, $c(e_2) = 4$, $c(e_3) = 12$, $c(e_4) = 5$, $c(e_5) = 3$, $c(e_6) = 10$, $c(e_7) = 7$, $c(e_8) = 10$.
Onda sa N(G,s,t,c) definiramo mrežu prikazanu na Slici 1.4.

[4]

Poglavlje 2

Maksimalni tok u mreži

U sljedećim poglavljima ćemo uvesti pojam minimalnog reza i potom predstaviti dvije metode za izračunavanje funkcije toka f za koju je ukupni tok F maksimalan. Koristit ćemo se radovima [4] i [10].

2.1 Minimalni rez i maksimalni tok u mreži

Neka je skup $S \subset V$, i neka je $\overline{S} = V \setminus S$. U daljnjem tekstu ćemo raspravljati o skupovima S, takvima da je $s \in S$ i $t \in \overline{S}$. Također, neka $(S;\overline{S})$ označava skup bridova koji su usmjereni iz čvorova u S prema čvorovima u $\overline{S}$. Ovaj se skup naziva rez prema naprijed. Skup $(\overline{S};S)$ se definira na sličan način i naziva se rez prema natrag. Unija skupova $(S;\overline{S})$ i $(\overline{S};S)$ se naziva rez definiran sa S.

Prema definiciji ukupan tok F se mjeri u ponoru. Naš sljedeći cilj je pokazati da se F može mjeriti na bilo kojem rezu.

Lema 2.1.1. *Za svaki $\{s\} \subset S \subset (V \setminus \{t\})$ i svaku funkciju toka f ukupnog toka F vrijedi:*

$$F = \sum_{e \in (S;\overline{S})} f(e) - \sum_{e \in (\overline{S};S)} f(e). \tag{2.1}$$

Dokaz. Prema pravilu čvora, za svaki $v \in V \setminus \{s,t\}$,

$$0 = \sum_{e \in \alpha(v)} f(e) - \sum_{e \in \beta(v)} f(e). \tag{2.2}$$

Uzimajući u obzir (1.1):

$$F = \sum_{e \in \alpha(t)} f(e) - \sum_{e \in \beta(t)} f(e). \tag{2.3}$$

Zbrajanjem jednadžbi na lijevoj strani dobijemo F.

Kako bi vidjeli šta se događa na desnoj strani jednadžbe pogledajmo najprije neki proizvoljni brid $v_i \xrightarrow{e} v_j$.

Ako su i v_i i v_j elementi S onda se $f(e)$ ne pojavljuje na desnoj strani jednadžbe, što je u skladu sa (2.1).

Ako su i v_i i v_j elementi $\overline{S}$, onda se $f(e)$ pojavljuje dva puta. Jedan puta sa pozitivnim i jedan puta sa negativnim predznakom. Pojavljivanja se poništavaju, dakle i to je u skladu sa (2.1).

Ako je $v_i \in S$ i $v_j \in \overline{S}$, onda se $f(e)$ pojeavljuje kao dio jednadžbe (2.2) za v_j, sa pozitivnim predznakom i nigdje drugdje za ostale čvorove. U tom slučaju je $e \in (S\,;\overline{S})$, što je u skladu sa (2.1).

Napokon, ako je $v_i \in \overline{S}$ i $v_j \in S$, onda se $f(e)$ pojavljuje sa negativnim predznakom na desnoj strani jednadžbe kao dio (2.2) što je opet u skladu sa (2.1), jer je $e \in (\overline{S}\,; S)$.

□

Označimo sa $c(S)$ kapacitet reza određen sa S, koji definiramo sa:

$$c(S) = \sum_{e \in (S;\overline{S})} c(e). \tag{2.4}$$

Lema 2.1.2. *Za svaku funkciju toka f, sa ukupnim tokom F, i svaki $\{s\} \subset S \subset (V \setminus \{t\})$ vrijedi sljedeća nejednakost:*

$$F \leq c(S). \tag{2.5}$$

Dokaz. Iz leme (2.1.1) slijedi:

$$F = \sum_{e \in (S;\overline{S})} f(e) - \sum_{e \in (\overline{S};S)} f(e). \tag{2.6}$$

Prema pravilu brida, za svaki brid e, $0 \leq f(e) \leq c(e)$. Slijedi da,

$$F \leq \sum_{e \in (S;\overline{S})} c(e) - 0. \tag{2.7}$$

Dakle prema (2.4) vrijedi $F \leq c(S)$.

□

Sljedeći Korolar je jako bitan i slijedi iz Leme (2.1.2), a pomaže nam da dokažemo da ako je dani ukupni tok F maksimalan, onda je kapacitiet pripadajućeg reza definiranog sa S minimalan.

Korolar 2.1.3. *Ako F i S zadovoljavaju jednadžbu* (2.5) *u jednakosti, onda je F maksimalan i rez definiran sa S je minimalnog kapaciteta.*

2.2 Ford-Fulkersonov algoritam

Ford i Fulkerson [7] predlažu upotrebu *proširenih puteva* za promjenu dane funkcije toka f radi povećanja ukupnog toka. Koristi se postupak za pronalaženje proširenog puta, ako on postoji. Ako je prošireni put pronađen, tada ga koristimo za povećanje ukupnog toka. Ako ga nema, trenutni tok je maksimalan i proces se prekida.

Ako zanemarimo smjer bridova, prošireni put P je obični usmjereni put iz s u t. Obično dodajemo $\Delta > 0$ ukupnom toku dodavanjem putu P Δ dodatnih jedinica toka. Da bi to napravili, ako je brid e u P usmjeren u smjeru iz s prema t, mora vrijediti $c(e) \geq f(e) + \Delta$. Ako je brid e od P usmjeren u suprotnom smjeru, moramo moći smanjiti put s $f(e)$ na $f(e) - \Delta$. Prema tome, mora vrijediti $f(e) \geq \Delta$.

U pokušaju da pronađemo prošireni put za dani tok, koristi se postupak označavanja. Prvo označimo s. Nadalje, dok god postoji neoznačen čvor v za koji je pronađen proširujući put iz s u v, v postaje označen. Ako je t označen, tada je pronađen proširujući put i proces označavanja se prekida.

Svakom čvoru v dodijelimo oznaku $\lambda(v)$, za koju vrijedi jedno od sljedećeg:

- $\lambda(s) = \infty$; jedino čvor s ima ovu oznaku, i to je zapravo jedina oznaka koju može imati
- $\lambda(v) = NIL$; u ovom slučaju kažemo da v nema oznaku
- $\lambda(v) = (e, \sigma)$, gdje je e brid kroz koji je dodijeljena oznaka čvoru v, $\sigma = +$ ako je e korišten prema naprijed i $\sigma = -$ ako je e korišten prema natrag.

Kažemo da je brid *u—v koristan* iz u prema v ako je $\lambda(u) \neq NIL$, $\lambda(v) = NIL$, i vrijedi jedan od sljedećih uvjeta:

- $u \xrightarrow{e} v$ i $f(e) < c(e)$. U ovom slučaju,kažemo da je *e korisan prema naprijed*.
- $v \xrightarrow{e} u$ i $f(e) > 0$. U ovom slučaju,kažemo da je *e korisan prema natrag*.

Algoritam 2.1. opisuje postupak označavanja, nazvan OZNACI. Ulazni podaci algoritma sastoje se od mreže N i postojeće funkcije toka f. $\lambda(\cdot)$ je definiran za sve čvorove. U postupku se ove oznake mogu izmijeniti. Nadalje, $\lambda(\cdot)$ je i ulazni i izlazni podatak. Izlaz OZNACI-a sadrži funkciju $\Delta(\cdot)$ koja je definirana na podskupu od E i njena vrijednost je pozitivan realan broj.

Algoritam OZNACI (N, f, λ, Δ)
while postoji brid $u \xrightarrow{e} v$ koji je koristan iz u u v i $\lambda(t) = NIL$ *do*
    if e je koristan prema naprijed *then do*
        $\lambda(v) \leftarrow (e, +)$
        $\Delta(e) \leftarrow c(e) - f(e)$
    if e je koristan prema natrag *then do*
        $\lambda(v) \leftarrow (e, -)$
        $\Delta(e) \leftarrow f(e)$

Algoritam 2.1.:Postupak označavanja

Algoritam PROSIRI (G, λ, Δ, f)

isprazni Q
$\Delta \leftarrow \infty$
$v \leftarrow t$
while $v \neq s$ *do*
    Q $\leftarrow \lambda(v)$
    neka $\lambda(v) = (e, \sigma)$
    $\Delta \leftarrow min\{\Delta, \Delta(e)\}$
    neka e povezuje u i v
    $v \leftarrow u$
while Q nije prazen
    remove prvu oznaku, (e, σ) iz Q
    if $\sigma = +$ *then do*
        $f(e) \leftarrow f(e) + \Delta$
    else $(\sigma = -)$ *do*
        $f(e) \leftarrow f(e) - \Delta$

Algoritam 2.2.:Postupak povećavanja toka

Ako je $\lambda(t) \neq NIL$ kada OZNACI završi, proširujući put postoji. Povećani put je nađen u PROSIRI proceduri i koristi se za promjenu funkcije f i povećanje F. To je pokazano u Algoritmu 2.2.

Ulazni podaci PROSIRI-a sastoje se od $G(V, E)$, $\lambda(\cdot)$ i $\Delta(\cdot)$. Funkcija toka f je ulazni i izlazni podatak. Skup oznaka čvorova Q i realni broj Δ su unutrašnje varijable.

Ford-Fulkersonov postupak je opisan u Algoritmu 2.3. i koristi OZNACI i PROSIRI kao potprograme. U početku, f je bilo koji tok u mreži N i u nedostatku boljih ideja,

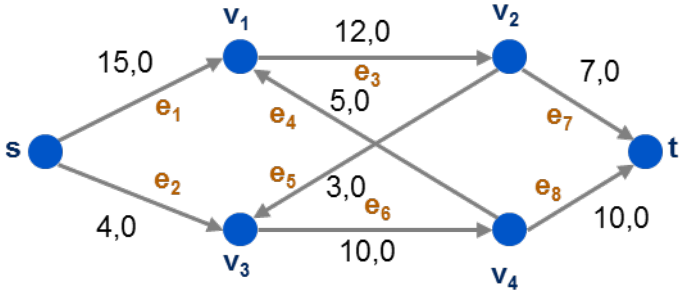
možemo uzeti $f(e) = 0$ za svaki brid e. Prema tome, ulazni podatak je N, a f je ulazni i izlazni podatak.

Primjetimo da ako za brid $e \in \alpha(s)$ u početku vrijedi $f(e) = 0$, tada se $f(e)$ nikad ne mijenja. Analogno i za brid $e \in \beta(t)$.

Algoritam FORD-FULKERSON (N,f)

for every $v \in V$ *do*
    $\lambda \leftarrow NIL$
$\lambda(s) \leftarrow \infty$
call OZNACI (N,f,λ, Δ)
while $\lambda(t) \neq NIL$ *do*
    call PROSIRI (G, λ, Δ, f)
    for every $v \in V \setminus \{s\}$ *do*
        $\lambda(v) \leftarrow NIL$
    call OZNACI (N, f, λ, Δ)
print "Trenutni tok je maksimalan"

Algoritam 2.3.:Ford-Fulkersonov algoritam

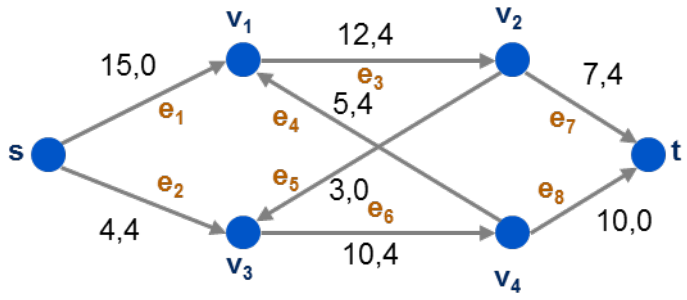


Slika 2.1: Primjer mreže

Kao primjer, promotrimo mreže prikazane na Slici 2.1. Kraj svakog brida e napišemo $c(e)$, $f(e)$. Pretpostavimo da je svugdje početni tok 0. U prvom valu raspodjela oznaka može biti ovakva: s je označen, e_2 koristimo za označavanje v_3, e_6 za označavanje v_4, e_4 za označavanje v_1, e_3 za v_2 i konačno e_7 za t. Put je

$$s \xrightarrow{e_2} v_3 \xrightarrow{e_6} v_4 \xrightarrow{e_4} v_1 \xrightarrow{e_3} v_2 \xrightarrow{e_7} t,$$

$\Delta = 4$, i novi tok je prikazan na Slici 2.2.

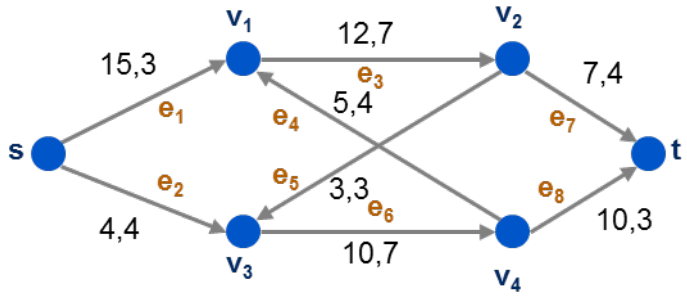


Slika 2.2: Prvi prošireni put

Drugi proširujući put može biti

$$s \xrightarrow{e_1} v_1 \xrightarrow{e_3} v_2 \xrightarrow{e_5} v_3 \xrightarrow{e_6} v_4 \xrightarrow{e_8} t,$$

$\Delta = 3$, i novi tok je prikazan na Slici 2.3.



Slika 2.3: Drugi prošireni put

Treći proširujući put može biti

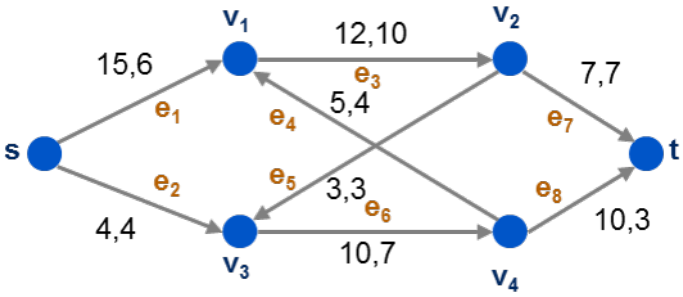
$$s \xrightarrow{e_1} v_1 \xrightarrow{e_3} v_2 \xrightarrow{e_7} t,$$

$\Delta = 3$, i novi tok je prikazan na Slici 2.4. Sve do sad koristilo se jedino označavanje prema naprijed. U sljedećoj primjeni OZNACI, još uvijek možemo označiti čvor v_1 preko e_1; i čvor v_2 preko e_3, od kojih su oba označena prema naprijed, ali takvog daljnjeg označavanja

nema. Međutim, e_4 je koristan prema natrag i pomoću njega možemo označiti v_4. Sada iskoristimo e_8 za označiti t. Proširujući put je

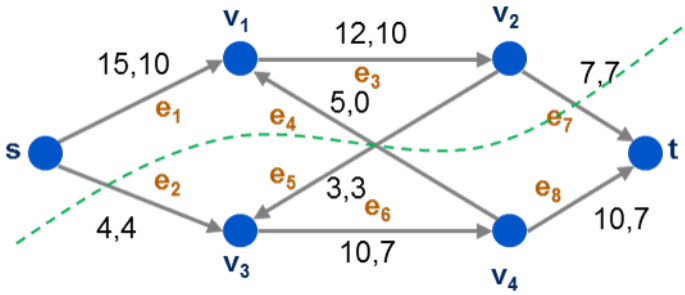
$$s \xrightarrow{e_1} v_1 \xleftarrow{e_4} v_4 \xrightarrow{e_8} t,$$

$\Delta = 4$, i novi tok je prikazan na Slici 2.5.



Slika 2.4: Treći prošireni put

Sada, ukupni tok, F, je jednak 14. Sljedeća primjena OZNACI ne doseže t. Skup označenih čvorova S je $\{s, v_1, v_2\}$ i rez prema unaprijed, $(S; \overline{S})$, sadrži bridove e_2, e_5 i e_7. Svi ovi bridovi su zasićeni, tj. $f(e) = c(e)$. Rez prema unatrag, $(\overline{S}; S)$, sadrži jedan brid e_4 i njegov tok je 0. Prema tome, $F = c(S)$, i po Korolaru 2.1.3, F je maksimalan i c(S) je minimalan.



Slika 2.5: Četvrti prošireni put

Lako je vidjeti da tok nastao ovim algoritmom ostaje korektan. Definicije $\Delta(\cdot)$ i Δ garantiraju da bridovi usmjereni prema naprijed neće premašiti $c(e)$, tj. $f(e) \leq c(e)$, i vrijedi $f(e) \geq 0$. Također, dok Δ jedinica tvore putanju od s do t, dolazni tok će ostati jednak odlaznom u svakom čvoru $v \in V \setminus \{s, t\}$.

Ako pretpostavimo da se dogodio zastoj u Ford-Fulkersonovoj proceduri, posljednji postupak označavanja nije dosegao t. Kao i gore, neka je S skup označenih čvorova u zadnjoj primjeni postupka označavanja. Ako je $e \in (S;\overline{S})$, tada je $f(e) = c(e)$, kako e nije koristan prema naprijed. Ako je $e \in (\overline{S};S)$,tada je $f(e) = 0$, kako e nije koristan prema natrag. Prema Lemi 2.1.1,

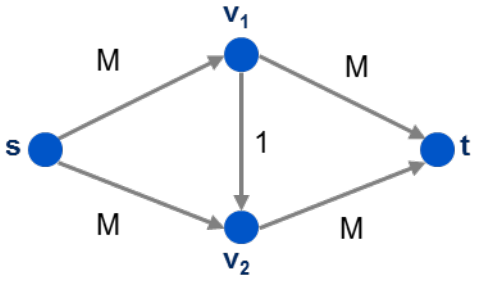
$$F = \sum_{e\in(S;\overline{S})} f(e) - \sum_{e\in(\overline{S};S)} f(e) = \sum_{e\in(S;\overline{S})} c(e) - \sum_{e\in(\overline{S};S)} 0 = c(S).$$

Prema Korolaru 2.1.3, F je maksimalan i c(S) je minimalan.

Preostaje nam razmotriti pitanje hoće li uvijek doći do zastoja u Ford-Fulkersonovoj proceduri. Prvo primjetimo vrlo bitnu značajku procedure: Ako je početni tok cjelobrojan, npr. 0 svugdje, i ako su svi kapaciteti cjelobrojni, tada se u algoritmu nikada ne pojavljuju razlomci. Algoritam zbraja i oduzima, ali nikad ne dijeli. Također, ako t je označen, nastala proširujuća putanja se koristi za povećanje funkcije toka za bar jednu jedinicu. Dok god postoji gornja granica za ukupni tok, proces se mora zaustaviti.

Ford i Fulkerson su pokazali da njihova procedura može pogriješiti ako kapaciteti mogu postati iracionalni brojevi. Njihov protuprimjer pokazuje beskonačan niz povećanja toka. Tok konvergira (u beskonačno mnogo koraka) vrijednosti koja je 1/4 maksimalnog ukupnog toka. Ovdje nećemo pokazivati taj primjer jer je prekompleksan, ali može se naći u [7].

Možemo tvrditi da za sve praktične primjene, možemo uzeti da će se algoritam sigurno zaustaviti. Ovo slijedi iz činjenice da se naši izračuni obično svode na fiksne baze prikaza broja (decimalni, binarni, itd.) s ograničenjima u broju znamenaka. Ipak, jednostavan primjer pokazuje slabost ovog argumenta. Razmotrimo mrežu prikazanu na Slici 5.6.



Slika 2.6: Primjer neefikasnosti Ford-Fulkerson algoritma

Pretpostavimo da je M neki veliki cijeli broj. Ako algoritam počinje s $f(e) = 0$ za svaki e, i naizmjenično koristi

$$s \to v_1 \to v_2 \to t$$

i

$$s \rightarrow v_2 \leftarrow v_1 \rightarrow t$$

kao povećani put, koristiti će $2M$ povećanja prije nego je $F = 2M$ postignuto. Ovo je eksponencijalno vrijeme s obzitom na duljinu ulaznih podataka, jer treba samo $\lceil \log_2(M + 1) \rceil$ bitova za prikazati M.

Edmonds i Karp [3] su prvi premostili ovaj problem. Pokazali su da ako koristimo BFS[1] u označavanju čime je svaki povećani put najkraći s obzorom na broj bridova, algoritam završava u $O(|V| \cdot |E|^2)$ vremenu, bez obzira na kapacitete. (Ovdje naravno pretpostavimo da naše računalo može u jednom koraku baratati s bilo kojim realnim brojem.) U sljedećem poglavlju trebali bi pokazati puno napredniji Dinitzev algoritam čija je složenost $O(|V|^2 \cdot |E|)$. Značajno efikasniji algoritam su prdstavili Goldberg i Tarjan[8], ali taj algoritam nećemo obraditi u ovom radu.

Postojanje algoritma Edmondsa i Karpa, ili Dinitza, osigurava da ako netko nastavi prema odgoovarajućoj strategiji postupka označavanja, algoritam se garantirano zaustavlja (u polinomijalnom vremenu). Kada se to dogodi, ukupni tok je maksimalan i rez je minimalan, čime se osigurava teorem o maksimalnom toku i minimalnom rezu:

Teorem 2.2.1. *U svakoj je mreži vrijednost maksimalnog toka jednaka kapacitetu minimalnog reza.*

2.3 Algoritam Dinitza

Kao u Ford-Fulkerson algoritmu, algoritam Dinitza počinje s nekom funkcijom toka f i želi ju poboljšati. Kada poboljšanje funkcije više nije moguće, algoritam se zaustavlja i ukupni tok F je maksimalan [1].

Neka je dana mreža $N(G(V, E), s, t, c)$ i funkcija toka f. Definirajmo drugu mrežu $N'(G'(V, E'), s, t, c')$ na sljedeći način:

- za svaki $e \in E$ takav da je $f(e) < c(e)$ vrijedi $e \in E'$. Također $c'(e) = c(e) - f(e)$. (Takav brid je *koristan naprijed* i $c'(e)$ je njen *rezidualni kapacitet*.)

- za svaki $e \in E$, $v_i \xrightarrow{e} v_j$ takav da je $f(e) > 0$, postoji brid $v_j \xrightarrow{e'} v_i$ iz E'. Također $c'(e) = f(e)$. (Takav brid zovemo *korisnom unatrag*.)

Primjetimo da ako je $0 < f(e) < c(e)$, tada e stvara dva paralelna brida suprotnog smjera u G', budući da je korisan u oba smjera. Prema tome, $|E| \leq |E'| \leq 2|E|$.

[1] Pretraživanje u širinu, detaljnju definiciju i povezane algoritme možete naću u [11]

Algoritam SLOJ (N', V_i, oznake vrhova, T)

$T \leftarrow \emptyset$
while postoji brid $u \rightarrow v$ u N', $u \in V_i$ i v nije označen, *do*
    $T \leftarrow T \cup \{v\}$
    označi v

Algoritam 2.4.:Pronalaženje sljedećeg sloja u BFS u N'

Prije nego krenemo na detaljan opis algoritma Dinitza, ovdje je sažetak njegove strukture:

Algoritam Dinitza radi u fazama. U svakoj fazi f se koristi za stvaranje odgovarajuće sporedne mreže N'. Slojevita mreža N'' stvara se BFS-om u N', počinjući iz izvora s. Ako ponor t nije dohvaćen, algoritam se zaustavlja i trenutni tok(protok) je maksimalan. Ako dohvatimo t, maksimalni tok f'' nalazimo u N''.Tok f je tada proširen koristeći f'' i ulazimo u novu fazu.

Sada ćemo pokazati detaljan opis algoritma Dinitza i kasnije raspraviti o njegovoj korektnosti i vremenskoj složenosti.

Konstrukcija N'' iz N' koristi BFS slojeve. Izrada slojeva je opisana u potprogramu SLOJEVI, koji rekurzivno koristi potprogram SLOJ opisan u Algoritmu 2.4. Ulazni podatak u SLOJ-u je sporedna mreža N'', prethodni sloj (skup čvorova) V_i i oznake čvorova. Skup označenih čvorova se može mijenjati dok je SLOJ pokrenut (jer neki dodatni čvorovi mogu postati označeni), i to je ujedno i izlazni podatak. Izlazni podatak T daje skup čvorova koji mogu postati sljedeći sloj.

Potprogram SLOJEVI je opisan u Algoritmu 2.5. SLOJEVI za ulazni podatak ima N'. Izlazni podatak je l, i ako je t dohvaćen, slojevi su navedeni u $V_0, V_1, \ldots, V_l$. Očito $l \leq n-1$.

Ako je t dohvaćen, *slojevita mreža* $N''(G''(V'', E''), s, t, c'')$ je konstruirana na sljedeći način:

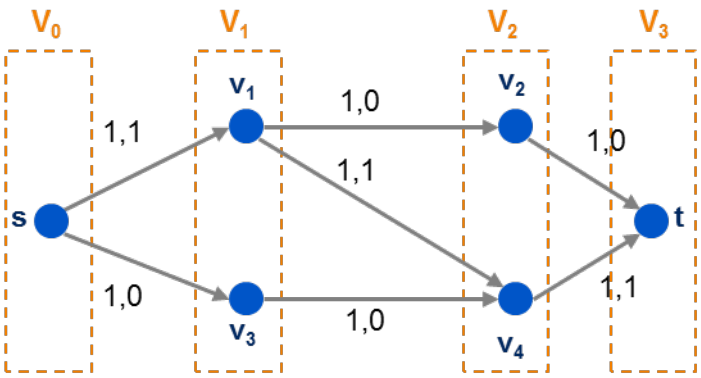
- $V'' = \bigcup_{i=0}^{l} V_i$,
- $E_i'' = \{u \xrightarrow{e''} v \mid u \in V_i, v \in V_{i+1}, e'' \in E'\}$,
- $E'' = \bigcup_{i=0}^{l-1} E_i''$,
- za svaki $e'' \in E''$, $c''(e'') = c'(e'')$.

Algoritam SLOJEVI $(N'; V_0, V_1, \ldots, V_{n-1}, \mathrm{l})$

Lokalne varijable: T je skup vrhova, oznake vrhova.
označi svaki $v \in V$ sa "neoznačen"
označi s
$V_0 \leftarrow \{s\}$
$i \leftarrow 0$
repeat
    call SLOJ $(N', V_i,$ "oznake vrhova", T)
    $i \leftarrow i + 1$
    $V_i \leftarrow T$
until $t \in T$ ili $T = \emptyset$
if $t \in T$ *then do*
    $l \leftarrow i$
    $V_i \leftarrow \{t\}$
else $(T = \emptyset)$ napiši: "f je maksimalni tok."

Algoritam 2.5.:Pronalaženja BFS Slojevi od N'

Dalje, konstruiramo *najveći tok ili blokirajući tok* u N''. Najveći tok f'' je tok u N'' takav da ne postoji povećani put duljine l u f''. Najveći tok ne mora biti maksimalan, kao što je prikazano na Slici 2.7. Ukupan tok je 1. Ovaj tok nije najveći, jer imamo tok ukupne veličine 2. Ipak on je maksimalan, budući da na svakom usmjerenom putu duljine 3 iz s u t, postoji barem jedan zasićen brid.



Slika 2.7: Primjer najvećeg toka koji nije maksimalan

Algoritam NADI-PUT (N", oznake blokiranja, S)

isprazni S
push (NIL,s) u S
while S nije prazan i t nije desni element u gornjem paru od S *do*
    neka je (x, v) gornji par od S
    if postoji neblokiran brid $v \xrightarrow{e'} v'$, *then do*
        $push$ (e', v') u S
    $else\ do$
        $if\ x \neq NIL\ then$ označi x sa 'blocked'
        pop (x, v) sa S

Algoritam 2.6.:Pronalaženje proširenog puta duljine l u N"

Lakše je naći maksimalan tok nego najveći, jer tok u bridovima nikad ne mora pasti. Drugim riječima, nije potrebno označavanje čvorova unatrag.

Procedura NAJVECI koja traži maksimalan tok u danoj slojevitoj mreži N'' počinje s $f''(e'') = 0$ za svaki $e'' \in E''$. Koristi dva potprograma: NADI-PUT i POVECAJ-TOK. U početku, svi bridovi označeni su s "unblocked". Kada brid postane zasićen, mijenja oznaku u "blocked". Također, ako zaključimo da tok u $u \xrightarrow{e''} v$ ne može narasti dok su svi putovi iz v u t blokirani, možemo promijeniti oznaku od e'' u "blocked".

NADI-PUT je opisan Algoritmom 2.6. Njegov ulazni podatak je N''. Blocking-oznake su i ulazni izlazni podatak, dok je stog S izlazni podatak. Procedura koristi DFS strategiju. Dok god postoji "unblocked" brid za nastaviti, idemo dalje i čuvamo niz bridova i njihovih krajnjih točaka u stogu S. Dakle, S je stog parova (e, v), gdje je v čvor u koji e ulazi. Ako je t dohvaćen, niz grana sačuvanih u S je direktan povećani put is s u t duljine l. Ako nema "unblocked" bridova izvan čvora koji smo dohvatili, uzmemo brid s vrha stoga S, promijenimo mu oznaku u "blocked" i vratimo se natrag na prethodni čvor puta. Ako smo zapeli na s, pretraga se prekida bez brida u S.

POVECAJ-TOK je opisan Algoritmom 2.7. Koristi dvije lokalne varijable: S' kao stog čvorova i Δ realni broj. N'' i S su ulazni podaci, dok su f'' i blocking-oznake i ulazni i izlani podaci. Povećani put sačuvan u S je pregledan dvaput. Tijekom prvog pregleda (Linije 3-6), minimalni rezidualni kapacitet brida povećanog puta je izračunat i sačuvan u Δ, i povećan put je vraćen u S'. U drugom pregledu, tok u svakom bridu povećanog puta je uvećan za Δ i ako bridovi postanu zasićeni, njihova oznaka se mijenja iz "blocked" u "unblocked". Očito, barem jedan brid puta postaje zasićen.

Algoritam POVECAJ-TOK (N", S, f",oznake blokiranja)

isprazni S'
$\Delta \leftarrow \infty$
while par na vrhu od S nije (NIL,s) *do*
    pop gornji par (e,v) sa S
    $\Delta \leftarrow min\{\Delta, c''(e) - f''(e)\}$
    push e u S'
while S' nije prazan *do*
    pop gornji brid e sa S'
    $f''(e) \leftarrow f''(e) + \Delta$
    $if\ f''(e) = c''(e)$ *then* označi e sa 'blocked'

Algoritam 2.7.:Povećaj tok na bridu proširenog puta u N"

NAJVECI, opisan algoritmom 2.8, nalazi najveći tok funkcije u slojevitoj mreži N''. Stog S i blocking-oznake su interne(unutarnje) varijable. Konačno, spremni smo pokazati algoritam Dinitza, opisan u algoritmu 2.9. Primjetimo da je ulazni podatak DINITZ-a mreža N, a izlaz je maksimalni tok f u N. Sve druge korištene varijable kao N', N'' i sve njihove komponente, uključujući l i f'' su interne(unutarnje) varijable.

Algoritam NAJVECI (N", f")

$for\ every\ e \in E''\ do$
    označi e sa 'blocked'
    $f''(e) \leftarrow 0$
run NADI-PUT(N",oznake blokiranja,S)
while S nije prazan *then do*
    run POVECAJ-TOK(N",S,f",oznake blokiranja)
    run NADI-PUT(N",oznake blokiranja,S)

Algoritam 2.8.:Konstruiraj maksimalni tok u slojevitoj mreži N"

Algoritam DINITZ (N;f)

for every $e \in E$ *do*
    $f(e) = 0$
$N' \leftarrow N$
run SLOJEVI $(N'; V_0, V_1, \ldots, V_{n-1},$ l)
while f nije označen kao maksimalni tok *do*
    konstruiraj N"
    run NAJVECI (N";f")
    for every $e' \in E''$ *do*
        if e' odgovara unaprijed orijentiranom bridu $e \in E$ *then do*
            $f(e) \leftarrow f(e) + f''(e)$
        else ($u \xrightarrow{e'} v \in N''$, ali $v \xrightarrow{e} u \in$) *do*
            $f(e) \leftarrow f(e) - f''(e')$
    konstruiraj N' iz (N,f)
    run SLOJEVI $(N'; V_0, V_1, \ldots, V_{n-1},$ l)

Algoritam 2.9.:Dinitz-ov algoritam

Lema 2.3.1. *Ako se procedura DINITZ zaustavi, nastala funkcija toka f je dopustivi tok u mreži N.*

Dokaz. Prvo, vidimo da u svakoj fazi funkcija toka f'' vrijedi u N''. Uvjeti za bridove vrijede jer je Δ u POVECAJ-TOK odabran na način da osigurava da nijedan brid nije prekapacitiran. Uvjeti za čvor vrijede jer je tok izgrađen proširujućim(povećanim) putevima. Iz definicije c', pa i c'', kada f izmijenimo dodavanjem(ili oduzimanjem) f'' prethodnoj f, uvjeti za bridove ostaju sačuvani u bridovima od G. Također, ako je f superpozicija dvaju tokova, koji se oba pridržavaju uvjeta za čvorove, to vrijedi i za njihovu sumu. □

Lema 2.3.2. *Ako se procedura DINITZ zaustavi, nastala funkcija toka f je maksimalni tok u N.*

Dokaz. Dokaz je sličan dokazu u Ford-Fulkersonovom algoritmu. Promotrimo zadnju fazu, u kojoj čvor t nije dohvaćen u proceduri SLOJEVI (Linija 13). Neka je S unija $V_1, \ldots, V_i$, gdje je V_i zadnji(neprazni) sloj prije nego naiđemo na $T = \emptyset$. Sada promotrimo rezove $(S; \overline{S})$ i $(\overline{S}; S)$ u G. Svaki brid $u \xrightarrow{e} v$ u $(S; \overline{S})$ je zasićen, ako nije e je korisno iz $u \in V_i$ prema v, pa T nije prazan. Također, za svaki brid $u \xrightarrow{e} v$ iz $(\overline{S}; S)$, vrijedi $f(e) = 0$ ili je e koristan iz $v \in V_i$ prema u, pa T nije prazan. Ostatak dokaza je identičan kao u slučaju Ford-Fulkersona. □

Lema 2.3.3. *U svakoj fazi, osim zadnje, tok f'' je najveći u N''.*

Dokaz. U početku, svi bridovi od N'' su "unblocked" (vidi Algoritam 2.8). Brid $u \xrightarrow{e} v$ ima oznaku "blocked" u dva slučaja:

- e je zasićen.
- Ne postoji "unblocked" brid bez v.

U svakom slučaju, kada brid ima oznaku "blocked", unaprijed je poznato da nije moguće dodatno proširenje puta. Dakle, kad su svi bridovi bez s blokiranog, tok f'' je najveći. Ovaj slučaj je otkriven procedurom FIND-PATH(vidi Algoritam 2.6) pomoću S koji je prazan. □

Lema 2.3.4. *Vrijeme trajanja svake faze je $O(|V| \cdot |E|)$.*

Dokaz. Vrijeme potrebno za konstrukciju N'' je $O(|E|)$, jer je to u suštini BFS na N'. Primjetimo da u Algoritmu 2.6 prelazimo bridove(vidi Linije 5-6). Ako se vratimo natrag na čvor koji nije s, brid označimo s "blocked". Broj uzastopnih prelazaka brida između ova dva blokiranja brida je ograničeno s l iz sljedećih razloga:

- Ako je l prelazaka izvedeno uzastopno bez vraćanja istim putem, tada je t dohvaćen i proširujući put je sačuvan u S. Procedura POVECAJ-TOK je pozvana i bar jedan brid postaje zasićen i označen s "blocked".
- Moguće je vratiti se natrag u čvor različit od s. U tom slučaju, u manje od l uzastopnih prelazaka ili se događa novo vraćanje ili je t dohvaćen, i kao u prethodnom slučaju barem jedan brid označen je s "blocked".

Kako je broj bridova u N'' reda $O(|E|)$ i kako svaki brid najviše jednom može biti "blocked", ukupan broj prelazaka kroz bridove je $O(|V| \cdot |E|)$. Nije teško vidjeti da je trajanje svih drugih operacija u proceduri NAJVECI ograničeno s konstantom puta broj prelazaka bridova. □

Primjetimo da je duljina slojevite mreže N'' označena s l. Kako moramo usporediti duljine slojevitih mreža u uzastopnim fazama, označimo s l_k duljinu slojevite mreže u k-toj fazi, $k \geq 1$.

Lema 2.3.5. *Ako $(k+1)$-va faza nije zadnja, tada je $l_{k+1} > l_k$.*

Dokaz. Neka je P put duljine l_{k+1} u slojevitoj mreži $(k+1)$-ve faze koji počinje u s i završava u t:

$$P : s = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \cdots v_{l_{k+1}-1} \xrightarrow{e_{l_{k+1}}} v_{l_{k+1}} = t.$$

Prvo, pretpostavimo da se svi članovi iz P pojavljuju u k-toj slojevitoj mreži. Neka je V_j j-ti sloj k-te slojevite mreže. Tvrdimo da ako je $v_a \in V_b$, tada je $a \geq b$. To se pokaže indukcijom po a. Za $a = 0 (v_0 = s)$, tvrdnja očito vrijedi. Pretpostavimo $v_{a+1} \in V_c$.

Prema pretpostavci indukcije, $(a \geq b)$, i ako $b + 1 \geq c$, tada je $a + 1 \geq b + 1 \geq c$, čime smo pokazali korak indukcije. Ako je $b + 1 < c$, tada brid e_{a+1} nije iskorišten u k-toj fazi jer nije ni u k-toj slojevitoj mreži, gdje se svi bridovi nalaze između susjednih slojeva. Ako e_{a+1} nije iskorišten u k-toj slojevtoj mreži i korisan je iz v_a prema v_{a+1} u početku faze $k + 1$, tada je korisn iz v_a prema v_{a+1} i u početku faze k. Dakle v_{a+1} ne može pripadati V_c jer po algoritmu pripada prethodnom sloju k-te mreže.

Posebno, $t = v_{l_{k+1}} \in V_k$. Stoga $l_{k+1} \geq l_k$. Također, jednakost ne može vrijediti jer u ovom slučaju, cijeli P je u k-toj slojevitoj mreži i ako su sve grane korisne u početku faze $k + 1$, tada konačni tok f'' faze k nije maksimalan. Ovo dokazuje tvrdnju leme u slučaju da su svi čvorovi od P u k-toj slojevitoj mreži.

Ako svi čvorovi od P nisu u k-toj slojevitoj mreži, neka je $v_a \xrightarrow{e_{a+1}} v_{a+1}$ prvi brid od P takva da su $b, v_a \in V_b$, ali v_{a+1} nije u k-toj slojevitoj mreži. Dakle, e_{a+1} nije korišten u fazi k. Kako je e_{a+1} koristan u početku faze $k + 1$, koristan je i u početku faze k. Jedini mogući razlog da v_{a+1} ne pripada V_{b+1} je da vrijedi $b + 1 = l_k$ i $v_{a+1} \neq t$. Kako je $t = v_{l_{k+1}}$, slijedi $a + 1 < l_{k+1}$. Prethodno smo pokazali $a \geq b$. Stoga $l_k = b + 1 \leq a + 1 < l_{k+1}$. □

Korolar 2.3.6. *Broj faza je ograničen s $|V|$.*

Dokaz. Očito $l_1 \geq 1$. Prema *Lemi 2.3.5*, ako k-ta faza nije zadnja vrijedi $l_k \geq k$. Kako je $k \leq l_k \leq |V| - 1$, broj faza, uključujući i zadnju, je ograničen s $|V|$. □

Teorem 2.3.7. *Algoritam Dinitza završava u vremenu reda $O(|V|^2|E|)$ i daje maksimalan tok.*

Dokaz. Prema *Korolaru 2.3.6*, broj faza je ograničen s $|V|$. Po *Lemi 2.3.4*, svaka faza zahtijeva $O(|V| \cdot |E|)$ vremena. Stoga cijeli algoritam zahtijeva $O(|V|^2 \cdot |E|)$ vremena do završetka. Prema *Lemi 2.3.2* rezultirjući tok je maksimalan. □

2.4 Mreže sa gornjim i donjim ograničenjima

U prijašnjim poglavljima smo pretpostavili da je tok svakog brida ograničen odozgo sa kapacitetom brida, a da je donje ograničenje na tok za svaki brid jednako 0. Ova pretpostavka je značajna jer je dodjeljinvanje $f(e)$ za svaki brid e definiralo korektan tok, i algoritam za povećavanje toka je mogao početi sa tim nula tokom.

U ovom poglavlju, ćemo osim gornje granice $c(e)$ pretpostaviti da tok ima i donju granicu $b(e)$. Tako da je pravilo brida potrebno izmjeniti na sljedeći način:

Tok $f(e)$, za svaki brid e, mora zadovoljiti

$$b(e) \leq f(e) \leq c(e)$$

Pravilo čvora ostaje isto.

Tako da se problem pronalaženja maksimalnog toka za mrežu $N(G(V, E), s, t, b, c)$ može podijeliti u dva dijela. Prvo, treba provjeriti da li N ima korektan tok, a ako je odgovor pozitivan, naći jedan takav tok. Drugo, povećati taj tok kako bi se našao maksimalni tok.

Sljedeću metodu testiranja da li dana mreža N ima korektnu funkciju toka su predstavili For i Fulkerson [7]. Ona pretvara problem u problem pronalaženja maksimalnog toka u *pomoćnoj mreži* $\tilde{N}(\tilde{G}(\tilde{V}, \tilde{E}), \tilde{s}, \tilde{t}, \tilde{b}, \tilde{c})$, u kojoj su sve donje granice jednake 0. Ovdje, $\tilde{s}$ i $\tilde{t}$ nisu više izvor i ponor i moraju zadovoljiti pravilo čvora. Sada ćemo pokazati da orginalna mreža N ima korektan tok, ako i samo ako maksimalni tok od $\tilde{N}$ zadovoljava jedan kriterij koji ćemo uskoro iskazati.

$\tilde{N}$ definiramo na sljedeći način:

- $\tilde{V} = V \cup \{\tilde{s}, \tilde{t}\}$, gdje su $\tilde{s}$ i $\tilde{t}$ dva nova čvora.
- $\tilde{E} = E \cup S \cup T \cup \{e', e''\}$, gdje su S T i $\{e', e''\}$ skupovi novih bridova definirani sa:

$$S = \{\tilde{s} \to v | v \in V\},$$

$$T = \{v \to \tilde{s} | v \in V\},$$

  i $s \xrightarrow{e'} t$ i $t \xrightarrow{e''} s$.

- $c : \tilde{E} \to R^+ \cup \{\infty\}$ je definirano posebno za svaki od četiri skupa bridova:
  - Za $e \in E$,
  $$\tilde{e} = c(e) - b(e).$$
  - Za $\tilde{s} \xrightarrow{\sigma} v, \tilde{c}(\sigma) = \sum_{e \in \alpha(v)} b(e)$.
  - Za $v \xrightarrow{\tau} \tilde{t}, \tilde{c}(\tau) = \sum_{e \in \beta(v)} b(e)$.
  - $\tilde{c}(e') = \infty$ i $\tilde{c}(e'') = \infty$

Sada možemo iskoristiti Dinitzov ili Ford i Fulkersonov algoritam da nađemo maksimalni tok u pomoćnoj mreži.

Teorem 2.4.1. *Orginalna mreža N ima korektan tok ako i samo ako maksimalni tok pomoćne mreže $\tilde{N}$ zasićuje sve bridove koji izlaze iz $\tilde{s}$, to jest sve bridove u S.*

Dokaz ovog teorema i detaljan postupak pronalaženja korektnog toka možete naći u [4], ali ih nećemo predstaviti u ovom radu.

Poglavlje 3

Kombinatorne primjene tehnika toka u mrežama

3.1 0-1 tok u mrežama

Nekoliko kombinatornih problema može biti rješeno pomoću tehnika toka u mrežama. U mrežama koje promatramo, kapacitet svih bridova je jednak 1. Da bi dobili bolje algoritme s manjom vremenskom složenosti moramo proučavati ove probleme tokova u mrežama. Ovdje pratimo Evenov i Tarjanov rad [6].

Razmotrimo sada problem maksimalnog toka gdje za svaki e iz $G(V, E)$ vrijedi $c(e) = 1$.

Prva primjedba je da u Dinitzovom algoritmu za maksimalni tok u slojevitoj mreži, svaki put kada pronađemo put, svi bridovi na njoj postanu blokirani; u slučaju da posljednji brid vodi u slijepu ulicu, vraćamo se natrag i ovoj brid postaje blokiran. Prema tome, ukupan broj prijelaza preko brida je ograničen s $|E|$ i cijela faza je vremenske složenosti $O(|E|)$. Pošto je broj faza ograničen s $|V|$, Dintzov algoritam za maksimalni tok je složenosti $O(|V| \cdot |E|)$.

Naš prvi cilj je pokazati da postoji bolja ograda vremenske složenosti: $O(|E|^{\frac{3}{2}})$. Međutim, prije toga trebamo pripremiti nekoliko rezultata.

Neka je $G(V, E)$ 0-1 mreža u kojoj je $c(e) = 1$ za svaki $e \in E$ i neka je f integralna funkcija toka. Definiramo $\tilde{G}(V, \tilde{E})$ na sljedeći način:

- Ako je $u \xrightarrow{e} v$ u G i $f(e) = 0$, onda je $e \in \tilde{E}$.
- Ako je $u \xleftarrow{e} v$ u G i $f(e) = 1$, onda je $u \xrightarrow{e'} v$ u $\tilde{G}$. Očito je e' nova grana koja je u korespondenciji s e.

Prema tome, $|E| = |\tilde{E}|$. Jasno, korisni bridovi slojevite mreže koje su konstruirane za G s postojećim tokom f, i sa svojim smjerom korisnosti, su i bridovi od $\tilde{G}$.

Označimo sa $(S;\overline{S})_G$, gdje je $s \in S, t \notin S$ i $\overline{S} = V - S$, skup bridova koji počinju iz čvoru u S i završavaju u čvoru u $\overline{S}$ i neka je $c(S,G)$ kapacitet odgovarajučeg reza u G. Također, neka je M ukupan maksimalni tok u G, dok je F ukupan postojeći tok.

Lema 3.1.1. *$\tilde{M} = M - F$.*

Dokaz. Neka je S podskup od V takav da je $s \in S$ i $t \notin$ iz S. Definicija od $\tilde{G}$ implicira da je

$$c(S,\tilde{G}) = |(S;\overline{S})_{\tilde{G}}| = \sum_{e\in(S;\overline{S})_G} (1 - f(e)) + \sum_{e\in(\overline{S};S)_G} f(e).$$

Međutim,

$$F = \sum_{e\in(S;\overline{S})_G} f(e) - \sum_{e\in(\overline{S};S)_G} f(e).$$

Dakle,

$$c(S,\tilde{G}) = |(S;\overline{S})_G| - F = c(S,G) - F.$$

Iz ovog slijedi da minimalni rez od G odgovara minimalnom rezu od $\tilde{G}$. Po max flow-min cut Teoremu 2.3.8 , kapacitet minimalnog reza od $\tilde{G}$ je $\tilde{M}$ (najveći ukupni tok u $\tilde{G}$) pa tvrdnja leme slijedi. □

Lema 3.1.2. *Duljina slojevite mreže za 0-1 mrežu definiranu s $G(V,E)$ (s danima s i t) i s tokom 0 posvuda je najviše $|E|/M$.*

Dokaz. Podsjećamo da je V_i skup čvorova i-tog sloja u slojevitoj mreži i E_i skup grana od V_{i-1} do V_i. Kako je $f(e) = 0$ za svaki $e \in E$, svi korisni smjerovi su prema naprijed. Prema tome, svaki E_i je jednak $(S;\overline{S})_G$ gdje je $S = V_0 \cup V_1 \cup \ldots \cup V_{i-1}$. Prema tome, po Lemi 2.1.1.

$$M \leq |E_i|.$$

Sumiravši za svaki $i = 1, 2, \ldots, k$ gdje je k duljina slojevite mreže, dobivamo $l \cdot M \leq |E|$, ili

$$l \leq |E|/M.$$

□

Teorem 3.1.3. *Za 0-1 mreže Dinitzov algoritam je vremenske složenosti $O(|E|^{\frac{3}{2}})$.*

Dokaz. Ako je $M \leq |E|^{\frac{1}{2}}$ onda je broj faza ograničen s $|E|^{\frac{1}{2}}$, pa teorem slijedi. Inače, promotrimo fazu tijekom koje ukupni tok doseže $M - |E|^{\frac{1}{2}}$. Ukupni tok F u $G(V,E)$ kada je za ovu fazu konstruirana slojevita mreža zadovoljava

$$F < M - |E|^{\frac{1}{2}}.$$

Ova slojevita mreža je identična onoj konstruiranoj za $\tilde{G}$ s tokom 0 posvuda. Prema tome, po *Lemi 3.1.1* vrijedi:

$$\tilde{M} = M - F > |E|^{\frac{1}{2}}.$$

Po *Lemi 3.1.2* duljina slojevite mreže l zadovoljava

$$l \leq |E|/\tilde{M} < |E|/|E|^{\frac{1}{2}} = |E|^{\frac{1}{2}}.$$

Dakle, broj faza do ovog trenutka je najviše $|E|^{\frac{1}{2}} - 1$, i kako je broj dodatnih faza do završetka najviše $|E|^{\frac{1}{2}}$, ukupan broj faza je najviše $2|E|^{\frac{1}{2}}$. □

0-1 mreža je *tipa* 1 ako nema paralelnih bridova. Za takvu mrežu možemo naći dodatnu gornju granicu za vremensku složenost. Prvo dokazujemo lemu sličnu *Lemi 3.1.2*.

Lema 3.1.4. *Neka $G(V, E)$ definira 0-1 mrežu tipa 1 s ukupnim maksimalnim tokom M od s do t. Duljina l prve slojevite mreže, kada je tok 0 posvuda, je najviše $2|V|/M^{\frac{1}{2}}$.*

Dokaz. Neka je V_i skup čvorova i-tog sloja. Kako nema paralelnih bridova, skup bridova E_{i+1}, od V_i do V_{i+1} u slojevitoj mreži zadovoljava $|E_{i+1}| \leq |V_i| \cdot |V_{i+1}|$ za svaki $i = 0, 1, \ldots, k-1$. Kako je svaki $|E_i|$ kapacitet reza, dobivamo da je

$$M \leq |V_i| \cdot |V_{i+1}|.$$

Prema tome, ili je $V_i \geq M^{\frac{1}{2}}$ ili $|V_{i+1}| \geq M^{\frac{1}{2}}$. Očito,

$$|V| \geq \sum_{i=0}^{l} |V_i| \geq \lfloor \frac{l+1}{2} \rfloor \cdot M^{\frac{1}{2}}.$$

Dakle,

$$\frac{|V|}{M^{\frac{1}{2}}} \geq \lfloor \frac{l+1}{2} \rfloor \geq \frac{l}{2},$$

i

$$\frac{2|V|}{M^{\frac{1}{2}}} \geq l,$$

pa lema slijedi. □

Teorem 3.1.5. *Za 0-1 mreže tipa 1, Dinitzov algoritam je vremenske složenosti $O(|V|^{\frac{2}{3}} \cdot |E|)$.*

Dokaz. Ako je $M \leq |V|^{\frac{2}{3}}$, rezultat slijedi odmah. Neka je F ukupni tok kada je slojevita mreža konstruirana za fazu tijekom koje ukupan tok dostiže vrijednost $M - |V|^{\frac{2}{3}}$. Ova slojevita mreža je identična onoj konstruiranoj za $\tilde{G}$ s tokom 0 posvuda. Kako $\tilde{G}$ može imati paralelne rubove, možda nije tipa 1, ali može imat najviše 2 paralelna brida od jednog

čvora do drugog; ako su e_1 i e_2 paralene i suprotnog smjera u G, $f(e_1) = 0$ i $f(e_2) = 1$, onda u $\tilde{G}$ imamo 2 paralelna brida: e_1 i e_2 . Rezultat sličan *Lemi 3.1.4* kaže da je

$$l < 2^{\frac{3}{2}}|V|/\tilde{M}^{\frac{1}{2}}.$$

Kako je $\tilde{M} = M - F > M - (M - |V|^{\frac{2}{3}}) = |V|^{\frac{2}{3}}$, dobijemo

$$l < \frac{2^{\frac{3}{2}}|V|}{|V|^{\frac{1}{3}}} = 2^{\frac{3}{2}} \cdot |V|^{\frac{2}{3}}.$$

Prema tome, broj faza do ovog trenutka je $O(|V|^{\frac{2}{3}})$. Kako je broj faza odavde do završetka najviše $|V|^{\frac{2}{3}}$, ukupan broj faza je $O(|V|^{\frac{2}{3}})$. □

U određenim primjenama, mreže koje nastaju zadovoljavaju uvjet da za svaki vrh različit od s i t, postoji ili samo jedan brid koji odlazi iz tog čvora ili jedan brid koji dolazi u taj čvor. Takve 0-1 mreže zovu se mreže *tipa* 2.

Lema 3.1.6. *Neka $G(V, E)$ definira 0-1 mrežu tipa 2 s ukupnim maksimalnim tokom M od s do t. Duljina l prve slojevite mreže, kada je tok 0 posvuda, je najviše* $(|V| - 2)/M + 1$.

Dokaz. Struktura od G implicira da se maksimalni tok u G može rastaviti na usmjerene putanje od s do t takve da nijedne od ovih putanja nemaju zajednički čvor osim zajedničkog početnog s i zajedničkog krajnjeg čvora t. (Tok može implicirati usmjerene kružne puteve koji su disjunktni u čvorovima i međusobno u putevima, osim možda u s ili t. Takvi kružni putevi nas ne zanimaju). Broj ovakvih puteva je M. Neka je λ duljina najkraćeg od ovih puteva. Prema tome, svaki put koristi barem $\lambda - 1$ unutarnjih čvorova. Imamo

$$M \cdot (\lambda - 1) \leq |V| - 2,$$

što implicira $\lambda \leq (|V| - 2)/M + 1$. U svakom slučaju, $l \leq \lambda$. Dakle, lema slijedi. □

Lema 3.1.7. *Ako G definira 0-1 mrežu tipa 2, i ako je f postojeća funkcija toka, onda i odgovarajući $\tilde{G}$ definira 0-1 mrežu tipa 2.*

Dokaz. Jasno je da $\tilde{G}$ definira 0-1 mrežu. Preostaje pokazati da za svaki čvor v, postoji ili jedan brid koji odlazi iz v ili jedan brid koji dolazi u v. Ako nema toka kroz čvor v (po f), onda u $\tilde{G}$, v ima točno iste vezane bridove pa uvjet i dalje vrijedi. Ako je tok kroz v jednak 1 (jasno je da ne može biti više od jedan), pretpostavimo da brid e_1 dolazi u čvor v, a brid e_2 odlazi iz čvora v. U $\tilde{G}$, niti jedan od ova dva brida se ne pojavljuje, ali su dodani bridovi e_1' i e_2' koji su obrnutog smjera od e_1 i e_2. Ostali brodovi u G koji su povezani s v, ostaju netaknuti u $\tilde{G}$. Prema tome, broj dolaznih bridova u v i odlaznih bridova iz v ostaju isti. Kako je G tipa 2, onda je i $\tilde{G}$ tipa 2. □

Teorem 3.1.8. *Za 0-1 mrežu tipa 2, Dinitzov algoritam je vremenske složenosti* $O(|V|^{\frac{1}{2}} \cdot |E|)$.

Dokaz. Ako je $M \leq |V|^{\frac{1}{2}}$, onda je broj faza ograničen s $|V|^{\frac{1}{2}}$ i teorem slijedi. Inače, razmotrimo fazu tijekom koje ukupni tok dostiže $M - |V|^{\frac{1}{2}}$. Stoga, slojevita mreža za ovu fazu je konstruirana kada vrijedi $F < M - |V|^{\frac{1}{2}}$. Ova slojevita mreža je identična onoj konstruiranoj za $\tilde{G}$ s tokom 0 posvuda. Također, po *Lemi 3.1.7*, $\tilde{G}$ je tipa 2. Prema tome, po *Lemi 3.1.6*, duljina slojevite mreže l je najviše $(|V| - 2)/\tilde{M} + 1$. Sada je $\tilde{M} = M - F > M - (M - |V|^{\frac{1}{2}}) = |V|^{\frac{1}{2}}$. Slijedi

$$l \leq \frac{|V| - 2}{|V|^{\frac{1}{2}}} + 1 = O(|V|^{\frac{1}{2}}).$$

Stoga, broj faza do ovog trenutka je najviše $O(|V|^{\frac{1}{2}})$. Kako je broj faza do završetka najviše $|V|^{\frac{1}{2}}$, ukupan broj faza je najviše $O(|V|^{\frac{1}{2}})$. □

[4]

3.2 Maksimalno sparivanje u bipartitnim grafovima

Skup bridova M, grafa $G(V, E)$ bez petlji, zove se *sparivanje* ako je svaki čvor incidentan s najviše jednim bridom iz M. Problem pronalaska maksimalnog sparivanja prvi je riješio Edmonds [2]. Najbolji poznati rezultat su dali Even i Kariv [5], složenosti $O(|V|^{2.5})$. Ovi algoritmi su previše kompliciranu da bi ih prikazali, a i ne koriste tehnike mrežnog toka.

Lakši problem je naći maksimalno sparivanje u *bipartitnom* grafu. Bipartitni graf je graf za koji vrijedi $V = X \cup Y, X \cap Y = \emptyset$, i svaki brid grafa ima jedan kraj u X, a drugi u Y. Ovaj problem je poznat i kao *problem ženidbe*. Naći ćemo rješenje ovog problema pomoću mrežnog toka i pokazati da je njegova složenost $O(|V|^{1/2)} \cdot |E|)$. Hopcroft i Karp su bili prvi koji su ostvarili ovaj rezultat [9].

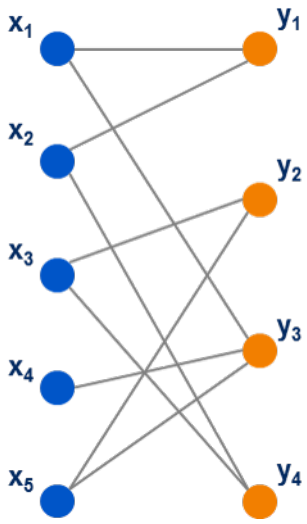
Rješavanje ovakvog problema ćemo pokazati kroz primjer bipartitnog grafa prikazanog na Slici 3.1.

Konstruiramo mrežu $N(G)$. Digraf $\bar{G}(\bar{V}, \bar{E})$ definiramo na sljedeći način:

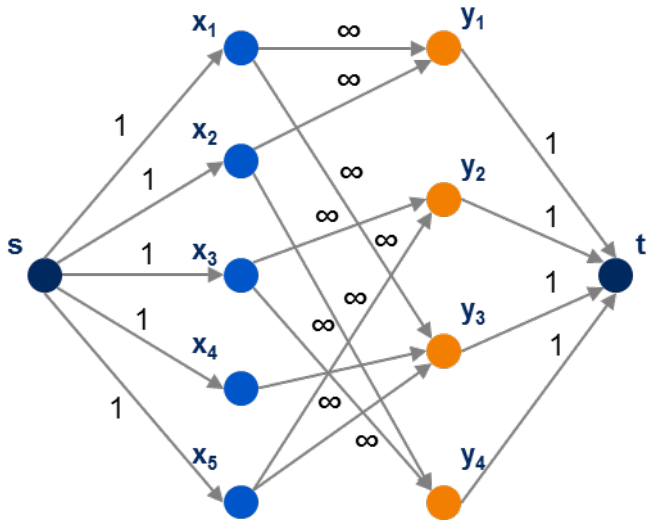
$$\bar{V} = \{s, t\} \cup V,$$

$$\bar{E} = \{s \to x \mid x \in X\} \cup \{y \to t \mid y \in Y\} \cup \{x \to y \mid x\text{—}y \text{ u } G\}.$$

Neka je $c(s \to x) = c(y \to t) = 1$ za svaki $x \in X$ i $y \in Y$. Također, za svaki brid $x \xrightarrow{e} y$, $c(e) = \infty$. (Ovaj beskonačni kapacitet je definiran u svrhu pojednostavljivanja dokaza Teorema 3.2.2. Kako postoji samo jedan brid koji ulazi u x, jediničnog kapaciteta, tok u $x \to y$ je ograničen s 1.) Izvor je označen sa s i ponor s t. Bipartitnom grafu na Slici



Slika 3.1: Primjer bipartitnog grafa

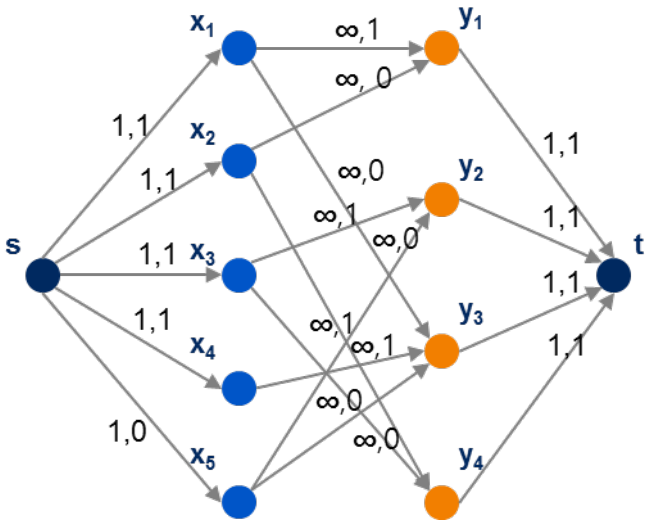


Slika 3.2: Pridružena mreža $N(G)$ bipartitinog grafa

3.1. je pridružena mreža $N(G)$ prikazana na slici 3.2.

Koristeći neki od algoritama za nalaženje maksimalnog toka (u našem primjeru Dinitzovog) pronalazimo maksimalni tok za konstruiranu pridruženu mrežu. Maksimalni tok

pridružene mreže bipartitnog grafa na Slici 3.1 je prikazan na Slici 3.3.



Slika 3.3: Maksimalni tok pridružene mreža $N(G)$ bipartitinog grafa

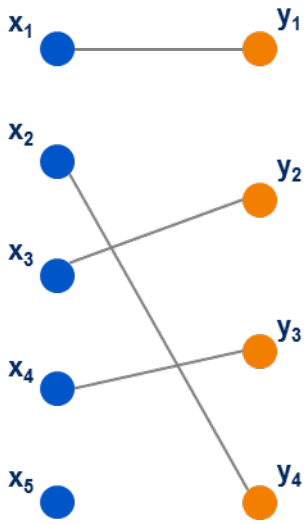
Eliminacijom bridova iz bipartitnog grafa na kojima u pridruženoj mreži maksimalnog toka nema protoka dolazimo do maksimalnog uparivanja. Maksimalno uparivanje dobiveno maksimalnim tokom sa Slike 3.3. je prikazano na slici 3.4.

Teorem 3.2.1. *Kardinalni broj maksimalnog sparivanja bipartitnog grafa G jednak je maksimalnom toku F u njemu pridruženoj mreži N(G).*

Dokaz. Neka je M maksimalno sparivanje. Za svaki brid $x \to y$ od M, koristimo usmjerenu putanju $s \to x \to y \to t$ za protok jedne jedinice od s do t. Očito, sve putanje su vršno-disjunktne. Stoga $F \geq |M|$.

Neka je f funkcija toka na $N(G)$, koja je cjelobrojna.(Ovdje nema gubitka općenitosti. Kao što smo vidjeli u Poglavlju 2, svaka mreža sa cjelobrojnim kapacitetima ima maksimalan cjelobrojan tok.) Sve usmjerene putanje koje povezuju s i t su oblika $s \to x \to y \to t$. Ako takvu putanju koristimo za tok iz s u t, onda nijedan drugi brid $x \to y'$ ili $x' \to y$ ne prenosi tok, kako postoji samo jedan brid $s \to x$ i njegov kapacitet je 1. Isto vrijedi i za $y \to t$. Slijedi da skup bridova $x \to y$, za koje je $f(x \to y) = 1$, tvori sparivanje u G. Dakle, $|M| \geq F$. □

Dokaz pokazuje kako rješenje mrežnog toka može doprinijeti maksimalnom sparivanju. U našem primjeru, maksimalni tok, nađen Dinitzovim algoritmom je prikazan na Slici 6.3(a), a njemu pridruženo sparivanje na Slici 6.3(b). Algoritam opisan u dokazu konstruira mreže tipa 2. Prema Teoremu 6.3, njegovo vrijeme trajanja je $O(|V|^{(}1/2) \cdot |E|)$.



Slika 3.4: Maksimalno uparivanje bipartitinog grafa dobiveno maksimalnim tokom pridružene mreže

Za svaki $A \subseteq X$, s $\Gamma(A)$ označimo skup svih čvorova u Y koji su povezani bridom s vrhom iz A. Sparivanje M je *potpuno* ako vrijedi $|M| = |X|$.

Teorem 3.2.2. *Bipartitni graf G ima potpuno sparivanje M ako i samo ako za svaki $A \subset X$ vrijedi $|\Gamma(A)| \geq |A|$.*

Dokaz. Očito, ako G ima potpuno sparivanje M, tada svaki x ima jedinstven "par" u Y. Dakle, za svaki $A \subset X$ vrijedi $|\Gamma(A)| \geq |A|$.

Pretpostavimo sada da G nema potpuno sparivanje. Neka je S skupo označenih čvorova (u Ford-Fulkersonovom ili Dinitzovom algoritmu) nakon prekida. Očito, ukupni maksimalni tok je jednak $|M|$, ali $|M| < |X|$. Neka je $A = x \cap S$. Kako su svi bridovi oblika $x \to y$ beskonačnog kapaciteta, $\Gamma(A) \subset S$. Također, nijedan čvor iz $Y \setminus \Gamma(A)$ nije označen jer ne postoji nijedan brid iz označenog čvora prema tom skupu čvorova. Sada imamo

$$(S\,;\bar{S}\,) = (\{s\}; X - A) \cup (\Gamma(A); \{t\}).$$

Kako je $(S\,;\bar{S}\,) = |M| < |X|$, vrijedi

$$|X - A| + |\Gamma(A)| < |X|,$$

što implicira $\Gamma(A)| < |A|$.

□

[4]

3.3 Problemi PERT usmjerenih grafova

Program Evalution and Review Tehnique, ili skraćeno PERT, je model upravljanja projektima. *PERT usmjereni graf* je konačni usmjereni graf $G(V, E)$ sa sljedećim svojstvima:

- Postoji čvor s koji se zove *početni čvor* i čvor $t(\neq s)$ koji se zove *prekidni čvor*.
- G nema usmjerenih ciklusa.
- Svaki čvor $v \in V \setminus \{s, t\}$ je na nekoj usmjerenoj putanji iz s u t.

PERT digraf interpretiramo na sljedeći način: Svaki brid predstavlja proces. Sjetimo se da $\alpha(v)$ označava skup bridova koji ulaze u v; $\beta(v)$ označava skup bridova koji izlaze iz v. Svi procesi koji su prikazani bridovima iz $\beta(s)$ mogu započeti odmah. Za svaki čvor v, proces prikazan bridovima iz $\beta(v)$ može započeti kada svi procesi prikazani bridovima iz $\alpha(v)$ završe.

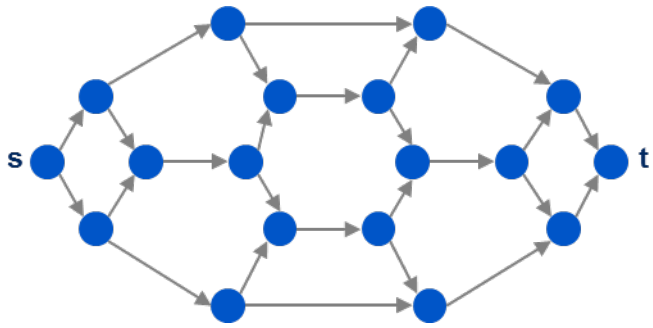
Naš prvi problem je koliko brzo cijeli projekt može biti gotov, tj. koje je najkraće vrijeme, od trenutka kada započinju procesi prikazani s $\beta(s)$ do trenutka kada su svi procesi prikazani s $\alpha(t)$ završeni. Pretpostavimo da su resursi za pokretanje procesa neograničeni. Kako bi dobro definirali ovaj problem, pretpostavimo da svaki $e \in E$ ima pridruženu *duljinu* $l(e)$, koja predstavlja vrijeme potrebno za izvršavanje procesa označenog s e. Minimalno vrijeme završetka možemo odrediti sljedećim algoritmom:

1. Pridružimo s oznaku 0 ($\lambda(s) \leftarrow 0$). Svi ostali čvorovi su "neoznačeni".
2. Pronađimo čvor v takav da je v neoznačen i svi bridovi iz $\alpha(v)$ izlaze iz označenih čvorova. Pridružimo
$$\lambda(v) \leftarrow \max_{e \in \alpha(v)} \{\lambda(u) + l(e)\}.$$
3. Ako je $v = t$, zaustavi; $\lambda(t)$ je minimalno vrijeme završetka. Inače, idi na Korak (2).

U Koraku (2), postojanje čvora v takvog da svi bridovi iz $\alpha(v)$ izlaze iz označenih čvorova osigurano je uvjetima (i) i (iii). Ako nijedan neoznačeni čvor ne zadovoljava uvjet, onda za svaki neoznačeni čvor v postoji nadolazeći brid koji izlazi iz nekog drugog neoznačenog čvora. Stalnim vraćanjem ovih bridova, jedan pronalazi usmjerenu putanju. Prema tome, ako nismo pronašli nijedan čvor, zaključujemo da (i) ili (ii) ne vrijede.

Indukcijom po redoslijedu označavanja lako se dokaže da je $\lambda(v)$ minimalno vrijeme u kojem svi procesi prikazani s $\alpha(v)$ moraju biti završeni.

Vremenska složenost može biti zadržana na $O(|E|)$ na sljedeći način:
za svaki čvor v brojimo bridove koji ulaze u njega iz neoznačenih čvorova; brojač je u početku postavljen na $d_{in}(v)$; svaki put kada čvor u postane označen, koristimo $\beta(u)$ da



Slika 3.5: Primjer PERT digrafa

smanjimo brojač za sve v takve da $u \to v$. Jednom kada brojač čvora v dosegne 0, on ulazi u niz čvorova koji se trebaju označiti.

Kada algoritam stane, vraćajući se iz s u t bridovima određenim oznakama čvorova, možemo nacrtati najdužu putanju iz s u t. Takva putanja se zove *kritična*. Očito, možemo imati više od jedne kritične putanje. Ako želimo skratiti vrijeme završetka $\lambda(t)$, onda moramo skratiti barem jedan brid na svakoj kritičnoj putanji.

Trebali bi razmotriti i problem duljine bridova u PERT usmjerenim grafovima. Pretpostavimo da svaki proces, prikazan bridovima, koristi jedan procesor za svoje izvršavanje. Koliko nam je procesora potrebno kako bi bili sigurni da izvršavanje neće biti odgođeno zbog nedostatka procesora? Želimo izbjeći odgodu bez oslanjanja na vrijednosti $l(e)$ jer su nepoznate i jer variraju s vremena na vrijeme.

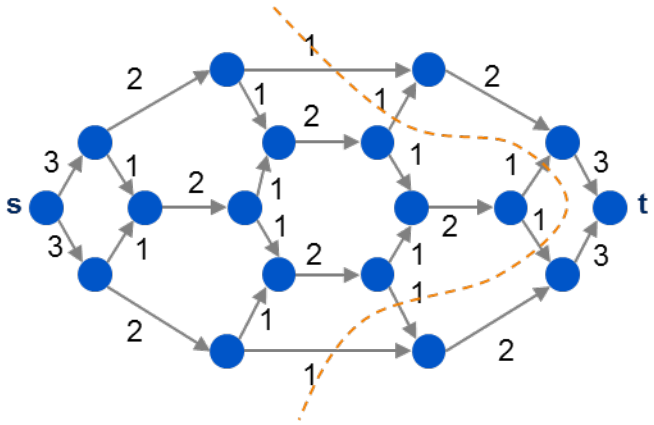
Riješimo problem minimalnog toka mreže čiji je digraf G, izvor s, ponor t, donja granica $b(e) = 1$ za sve $e \in E$ i nemamo gornju granicu ($c(e) = \infty$ za sve $e \in E$).

Promotrimo PERT usmjereni graf na Slici 3.5. Minimalni tok i maksimalni rez (koji su u ovom slučaju jedinstveni) su prikazani na Slici 3.6.

Skup bridova je *suglasan* ako za nijedna dva brida iz skupa ne postoji usmjerena putanja koja prolazi kroz njih. Sada, neka je T skup čvorova koji su označeni u prethodnom pokušaju pronalaska proširujuće putanje od t do s. Očito, $t \in T$ i $s \notin T$. Skup bridova $(\bar{T}; T)$ je maksimalan rez; u $(T; \bar{T})$ nema bridova; jer nema gornje granice za tok u bridovima i svaki takav brid omogućuje nastavak označavanja čvorova. Stoga, skup $(\bar{T}; T)$ je suglasan.

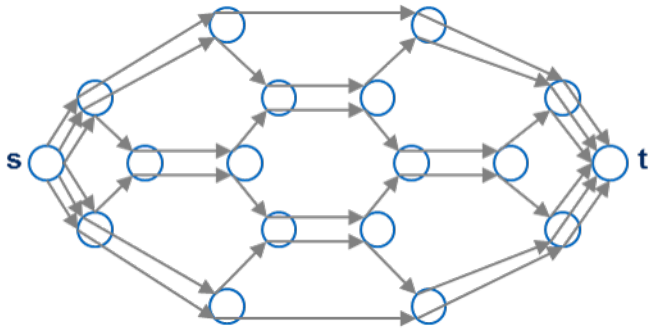
Ako je S skup concurrent bridova, onda je broj potrebnih procesora barem jednak $|S|$. Ovo možemo vidjeti dodijeljujući bridovima od S vrlo velike duljine i svim drugima male duljine. Kako nijedna usmjerena putanja ne vodi iz jednog brida od S do drugog, sve će biti u radu istovremeno. Ovo implicira da je broj potrebnih procesora barem jednak $|(\bar{T}; T)|$.

Ipak, tok može biti rastavljen na F usmjerenih putanja iz s do t, gdje je F ukupni minimalni tok, tako da je na svakom bridu barem jedan takav (kako je $f(e) \geq 1$ za svaki $e \in E$).



Slika 3.6: Minimalni tok PERT usmjereni graf

Ovo je prikazano primjerom na Slici 3.7. Sada svakom procesoru možemo pridružiti sve bridove jedne takve putanje. Svaki takav procesor izvršava procese predstavljene bridovima putanje, redom kojim se pojavljuju na putanji. Ako je jedan proces pridružen više od jednom procesoru, tada ga jedan od njih izvršava dok su drugi besposleni. Slijedi da kada god je proces koji odgovara $u \rightarrow v$ izvršiv, procesor kojem je dodijeljen ovaj proces je dostupan za njegovo izvršenje. Dakle, za naše potrebe je dovoljno F procesora.



Slika 3.7: Dekompozicija PERT usmjerenog grafa

Kako je $F = |(\bar{T}; T)|$, prema min-flow max-cut teoremu [1], broj na ovaj način dodijeljenih procesora je minimalan.

[1]Teorem analogan max-fow min-cut teoremu: Minimalna vrijednost toka od izvora s do ponora t u kapacitiranoj mreži sa nenegativnim donjim granicama jednaka je maksimalnom kapacitetu između svih rezova mreže.

Složenost ove procedure je sljedeća: možemo naći pravilan početni tok u vremenu $O(|V| \cdot |E|)$, crtanjem za svaki brid usmjerene putanje od s do t koja prolazi kroz taj brid. Ovu putanju pronalazimo počinjući iz brida, i nastavljajući naprijed ili natrag sve dok ne dostignemo s i t. Nadalje, riješimo problem maksimalnog toka, iz t u s. Dakle, složenost cijelog problema se svedi na složenost rješavanja jednog problema maksimalnog toka. [4] [11]

Poglavlje 4

Primjer primjene tokova u mrežama

U ovom ćemo poglavlju pokazati na nekoliko praktičnih primjera, kako koristeći tokove u mrežama riješiti odabrane kombinatorne probleme. Fokus će biti na problemima uparivanja dipartitnih grafova, a koristit ćemo algoritme Dinitz i Ford-Fulkerson kako bismo našli maksimalno uparivanje tj. maksimalni tok.

Uz pretpostavku da će oba algoritma uvijek dati optimalno rješenje problema, mjerit ćemo njihovu vremensku efikasnost tj. vrijeme potrebno da generiraju optimalno rješenje na istome računalu i problemu.

Za programiranje algoritama i generiranje rješenja na danim primjerima korišten je programski jezik *C#*, a računalo na kojem je testirana koristi Intel(R) Core(TM) i3-2120 CPU koji radi na 3.30 GHz. Računalo također ima 4 GB RAM-a i 64-bitini operativni sustav (Windows 10).

4.1 Raspodjela radnika

U nedostatku radnika poduzetnik Pero, vlasnik građevinske kompanije, se našao u problemu. Kako su u izgradnji potrebne različite vještine i znanja, koje nemaju svi njegovi radnici odlučio se unajmiti studenta računarstva da mu pomogne napraviti algoritam uz pomoću kojeg će moći postići optimalni razmještaj radnika po poslovima.

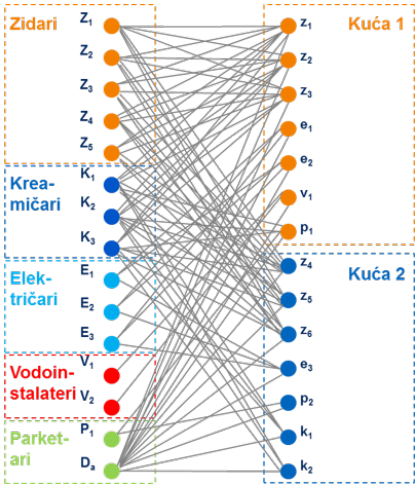
Koristeći poznanstava i kontakte došao je do Juraja, koji je završavao Magisterij Računarstva i Matematike na Zagrebačkom Sveučilištu. Juraj je kao i svaki savjesni student odmah prionuo poslu i predstavio rješenje koje se baziralo na kvalifikacijama radnika, gdje ih je algoritam razmještao na poslove za koje su imali potrebna znanja.

Juraj je očekivao da će algoritam zadovoljiti Peru, ali nakon prezentacije se pojavio veliki problem. Naime, Pero je tvrdio da bez obzira na kvalifikacije samo on zna koji je radnik u stanju raditi određeni posao. Postoje radnici koji nemaju kvalifikacije, ali imaju

veća znanja od nekih koji ih nemaju. Također, poslovi su nekada toliko jednostavni da ih može napraviti i neki radnik koji za njih nema napredna znanja.

Juraj je dugo razmišljao, i potom je došao na ideju da koristi tokove u mrežama kako bi našao optimalno uparivanje, omogućujući pritom Peri da sam odredi koji radnik je sposoban odraditi koji dati posao.

Kako bi testirao primjenu Juraj je odlučio pronaći najprije rješenje za trenutnu situaciju u kompaniji.



Slika 4.1: Dipartitini skupovi radnika i poslova

Kompanija ima 15 zaposlenika:

- 5 zidara koji rade opće i klasične zidarske poslove.
- 3 keramičara koji su specijalizirani za postavljanje pločica i zidnih obloga.Osim toga keramičari mogu raditi i sve opće i klasične zidarske poslove kao i zidari.
- 3 električara koji postavljaju elektro instalacije i rasvjetna tijela.
- 2 vodointstalatera za vodovodne instalacije i grijanje.
- 2 parketara za postavljanje parketa i podnih obloga.

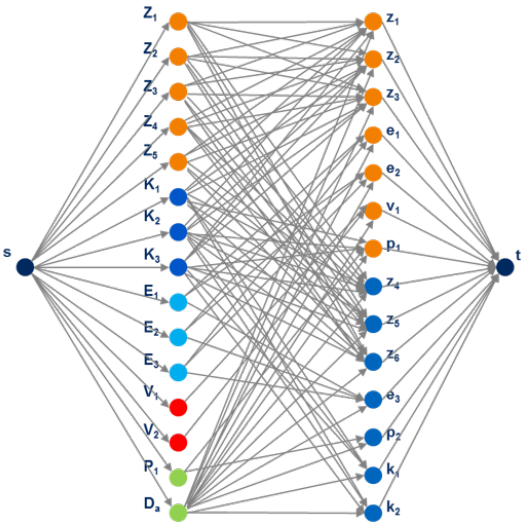
A za Damira, koji je parketar, Pero kaže da može raditi sve poslove, jer je u svojoj dugoj karijeri sve zanate već savladao.

Trenutno su u izgradnji dvije kuće:

- Za izgradnju prve su potrebna 3 zidara, 2 električara, vodoinstalater i parketar. Parketni posao je jednostavan tako da ga može raditi i jedan od keramičara.

- Za izgradnju druge su potrebna 3 zidara, električar, parketar i 2 keramičara.

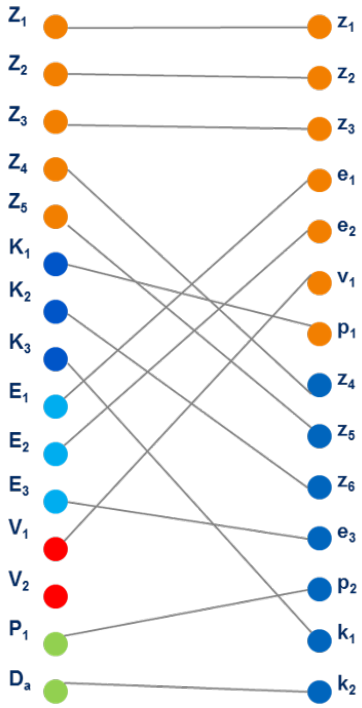
Juraj je nacrtao uparivanje dva skupa (Slika 4.1), te potom pripadajuću mrežu (Slika 4.2). Kapaciteti svih bridova u uparenoj mreži su 1.



Slika 4.2: Pridružena mreža raspodjeli radnika po poslovima

Za traženje maksimalnog toka je koristio Dinitzov algoritam. Pripadajuća optimalna mreža maksimalnog toka koji je dobio je prikazan na Slici 4.3.

Pero je bio toliko zadovoljan dobivenim rješenjem da je Juraja preporučio svim svojim kolegama, a Juraj je u samo nekoliko godina proširio posao i izvan granica Hrvatske te ostvario bogatstvo na aplikacijama za raspodjelu resursa.



Slika 4.3: Maksimalna raspodjela radnika po poslovima dobivena Dinitz-ovim algoritmom

4.2 Speed dating

Speed dating je koncept pronalaženja potencijalnih partnera u kojem muškarci i žene imaju priliku u kratkom vremenu upoznati velik broj potencijalnih partnera i odabrati one s kojima bi (na osnovi kratkog upoznavanja) htjeli ostvariti daljnji kontakt.

Inicijalno upoznavanje se događa u restoranu gdje svaka žena ima priliku provesti 5 minuta sa svakim muškarcem na tzv. kratkom spoju. Žene sjede svaka za svojim stolom i dodijeljen im je inicijalni partner za spoj. Nakon isteka 5 minuta oglasi se zvonce i muškarci se pomiču na sljedeći stol te upoznaju sljedeću potencijalnu partnericu. Postupak se ponavlja dok svi mogući parovi (tj. sve žene i svi muškarci) nisu imali priliku imati svoj kratki spoj.

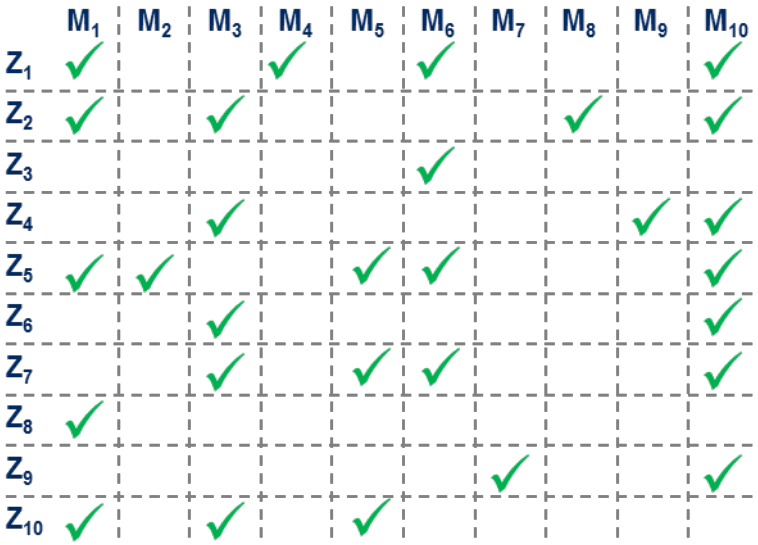
Za vrijeme događaja svi muškarci i ženem koriste kartice sa imenima kako bi označili sve one partnere s kojima bi htjeli ostvariti daljnji kontakt.

Vlasnik restorana u Zagrebu je, kako bi povećao popularnost svojeg restorana, odlučio prvi puta organizirati Speed dating događaj. Odlučio se oglasiti ga preko društvenih mreža i nakon pomnog odabira raznolikih kandidata je došao do konačnih 10 muškaraca i 10 žena koje je odlučio pozvati na upoznavanje. Kao dodatak događaju i kako bi potaknuo

konzumaciju jela i pića vlasnik je odlučio odmah nakon kratkih spojeva uparenim parovima ponuditi romantičnu večeru.

Zahvaljujući odličnoj pripremi sve je teklo prema planu: atmosfera je bila jako ugodna, svi potencijalni parovi su dobili priliku imati kratki spoj i pritom su ispunili svoje kartice. Vlasnik je sakupio kartice, i nenadano došao do neočekivanog problema. Nakon što je označio sve muškarce i žene koji su iskazali interes jedno za drugo, uvidio je da mu neće biti jednostavno utvrditi konačne parove za romantičnu večeru. Naime, neki muškarci i žene su bili izrazito uspješni i upareni su sa većim brojem partnera dok su neki zbog izbirljivosti ili nepopularnosti bili upareni sa samo jednom ili dvije osobe.

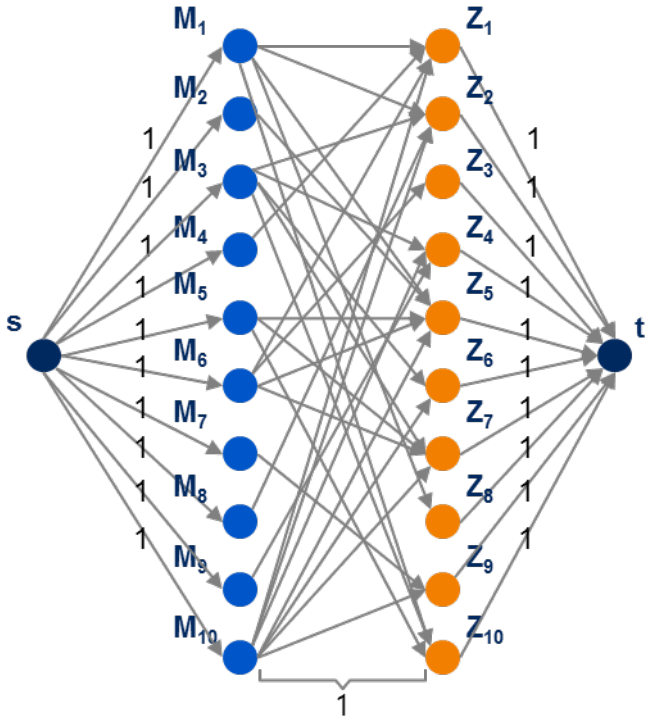
Tablica uparenih osoba je prikazana na Slici 4.4.



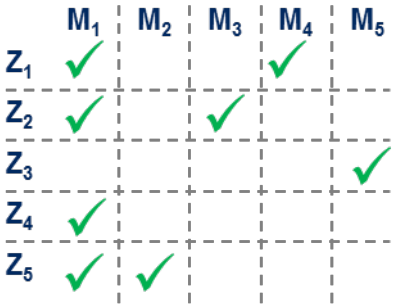
Slika 4.4: Tablica uparenih osoba

Srećom Vlasnik je prije nego se odlučio okušati u ugostiteljstvu magistrirao računarstvo i matematiku te se prisjetio jednog predavanja o primjeni tokova u mrežama za rješavanje ovakvih problema uparivanja dipartitnih skupova. Izvukao je svoju prašnjavu knjigu i našao dva algoritma pomoću kojih je mogao efikasno riješiti problem: algoritam Ford-Fulkerson i algoritam Dinitz. Kako nije znao koji je algoritam brži, a razmišljao je kako će mu isti ponovno trebati idući put kada bude organizirato Speed dating, odlučio se testirati oba i provjeriti koji će brže riješiti problem.

Najprije je konstruirao pridruženu mrežu i tokove prikazane na Slici 4.5. Kako nije bio siguran da li će i u budućnosti organizirati događaje za 20 ljudi, osim za trenutni problem odlučio ih je testirati i za slučaj da pozove 10 i 40 ljudi. Primjere koje je koristio su prikazani na Slikama 4.6. i 4.7.



Slika 4.5: Pridružena mreža tablici uparenih osoba

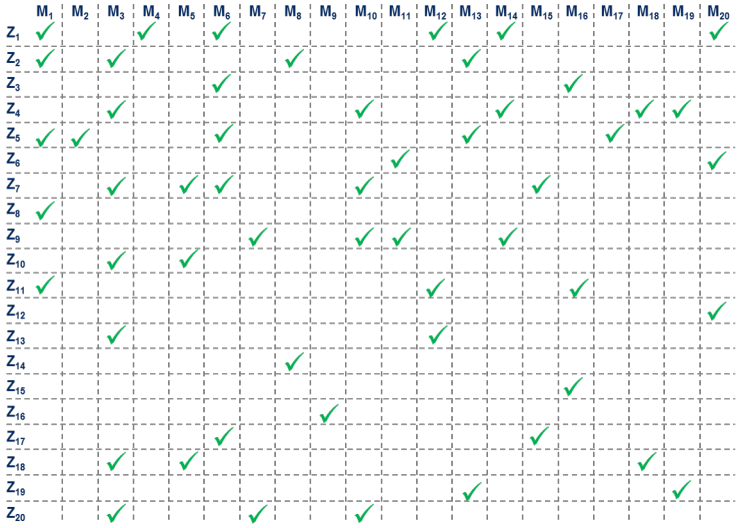


Slika 4.6: Tablica uparenih osoba za 10 osoba

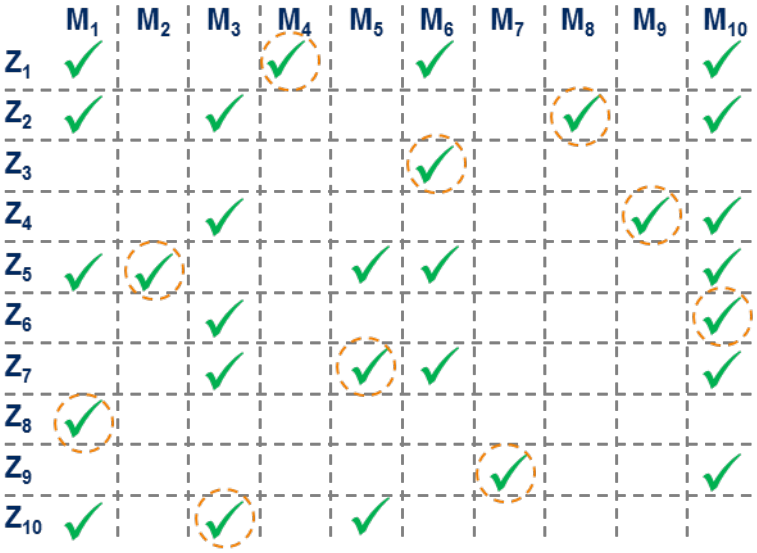
Dobiveno maksimalno uparivanje je potom označio krugovima na tablici uparenih osoba.

Kako bi bio siguran u dobiveni rezulatat svaki primjer je pokrenuo 4 puta i izračunao prosjek [1]. Nakon testiranja oba algoritma na primjerima za 10, 20 i 40 osoba dobio je

[1]Pokrenuo je program više puta jer je računalunalu zbog svoje arhitekture za tako kratka izvršavanja



Slika 4.7: Tablica uparenih osoba za 40 osoba



Slika 4.8: Konačni parovi koji će ići na romantičnu večeru

sljedeće rezultate:

Vlasnik restorana je u konačnici raspodijelio parove, a igrom slučaja postojala je ras-

potrebno svaki puta nešto drugačije vrijeme da izvrši isti program

	N=10	N=20	N=40
Ford-Fulkerson	10.440	12.251	13.401
	11.067	12.056	14.950
	10.700	11.216	13.410
	10.408	11.256	13.999
	10.654	11.695	13.940
Dinitz	0.746	0.764	1.635
	0.845	1.044	0.813
	0.673	0.701	0.851
	0.672	0.707	0.808
	0.734	0.804	1.027

Slika 4.9: Rezultati testiranja Ford-Fulkerson-ovog i Dinitz-ovog algoritma

podjela u kojoj su svi muškarci i sve žene uspjeli ostvariti romantičnu večeru sa jednom osobom koju su sami odabrali.

Na osnovi rezultata mjerenja vremenske efikasnosti Ford-Fulkerson-ovog i Dinitz-ovog algoritma Vlasnik je također zaključio da mu Dinitz-ov algoritam puno brži, tako da će njega koristiti u budućnosti za sljedeći speed dating.

Zaključak

U radu smo predstavili osnovne pojmove vezane za mreže i tokove u mrežama, kao i neke važne teoreme. Uveli smo dva osnovna algoritma za traženje maksimalnog toka u mrežama, od kojih Ford-Fulkerson predstavlja osnovni algoritam proširivanja tokova, dok Dinitzov algoritam predstavlja puno efikasniji i korišteniji primjer algoritma.

Potencijalno daljnje područje istraživanja bi bili noviji algoritmi, kao što su Edmonds i Karp [10], i Goldberg i Tarjan[11], te algoritami koji se koriste u ekonomiji, kao što je simpleks algoritam [12].

Kao primjere primjene tokova u mrežama u kombinatorici smo naveli probleme uparivanja dipartitih grfova te probleme PERT usmjerenih grafova koji se baziraju na jednom izvoru i jednom ponoru. Sljedeći korak bi bili problemi koji se baziraju na više izvora i ponora kao što su problemi Tutteovih matrica [11] i problem MTTM (Minimizacija troškova u mreži). [12]

Bibliografija

[1] E. A. Dinic, *Algorithm for Solution of Problem of Maximum Flow in Networks with Power Estimation*, Soviet Math. Dokl. (1970), br. 11, 1277–1280.

[2] J. Edmonds, *Paths, Trees, and Flowers*, Canadian J. of Math. **17** (1965), 449–467.

[3] J. Edmonds i R.M. Karp, *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems*, J. Assoc. Comput. Mach. (1972), br. 19, 248–264.

[4] S. Even, *Graph algorithms*, Cambridge University Press, 2011.

[5] S. Even i O. Kariv, *An $O(n^{2.5})$ Algorithm for Maximum Matching in General Graphs*, 16th Annual Symposium on Foundations of Computer Science, IEEE (1975), 100–112.

[6] S. Even i R. E. Tarjan, *Network Flown and Testing Connectivity*, SIAM J. on comput. **4** (1975), 507–518.

[7] L. R. Ford i D. R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[8] A. V. Goldberg i R. E. Tarjan, *A New Approach to Maximum Flow Problem*, J. Assoc. Comput. Mach. (1988), br. 35, 921–940.

[9] J. Hopcroft i R. M. Karp, *An $O(n^{5/2})$ Algorithm for Maximum Matching in Bipartite Graphs*, SIAM J. on comput. (1975), 225–231.

[10] D. Jungnickel i T. Schade, *Graphs, networks and algorithms*, Springer, 2005.

[11] B. Korte i J. Vygen, *Combinatorial Optimization, Theory and Algorithms*, Springer, 2002.

[12] Z. Lukač i L. Neralić, *Operacijska istraživanja*, Element, 2012.

Sažetak

Tema ovog rada je zanimljivo područje rješavanja kombinatornih problema pomoću tokova u mrežama.

U radu se najprije definiraju osnovni pojmovi i predstavljaju primjeri vezani za područja neusmjerenih i usmjerenih grafova, mreža i tokova u mrežama, kao što su bridovi, putevi, tokovi te kapacitet bridova. Nakon toga uvodi se centralni problem pronalaženja maksimalnog toka u danoj mreži. Potom dokazujemo da je problem ekvivalentan problemu minimalnog reza u danoj mreži čime dolazimo da najvažnijeg teorema tzv. Max-flow min-cut teorema.

Nakon što uvedemo potrebne pojmove i zakonitosti, predstaviti ćemo dva poznatija algoritma koji se koriste za pronalaženje maksimalnog toka u danim mrežama: Ford-Fulkerson algoritam i algoritam Dinitz. Oba algoritma kreću od nekog toka koji onda povećavaju dok ne dođu do maksimalnog toka. Ford-Fulkerson algoritam to postiže pomoću proširenih puteva, dok Dinitzov algoritam to postiže pomoću sporednih mreža i tzv. blokirajućih tokova.

Kako se dobar dio kombinatornih problema može prikazati i rješavati pomoću simplificirane verzije mreža sa 0-1 tokom, malo ćemo detaljnjije analizirati složenost Dinitzovog algoritma za tu vrstu mreža. Potom ćemo prikazati i objasniti nekoliko primjena mrežnih tokova za rješavanje određenih kombinatornih problema: maksimalno uparivanje u bipartitinim grafovima, te prebleme PERT usmjerenih grafova.

Za kraj ćemo na dva praktična problema maksimalnog uparivanja u bipartitnim grafovima: Raspodajele radika i Speed dating, pokazati kako bi se oni riješili pomoću tokova u mrežama, pritom koristeći i testirajući efikasnost Dinitzovog i Ford-Fulkersonovog algoritma koji su predstavljeni u prethodnim poglavljima.

Summary

The subject of this paper is the interesting topic of solving combinatorial problems using network flows.

The paper first defines the basic concepts and presents examples related to the areas of undirected and directed graphs, networks and network flows, as edges, paths, flows and edge capacity. After that, we introduce the central problem of finding the maximum flow in the given network. We prove that the problem is equivalent to the problem of the minimum cut in the given network, so that we come to the most important Max-flow min-cut theorem.

After introducing the necessary terms and principles, we will present two commonly used algorithms used to find the maximum flow in the given networks: Algorithms of Ford and Fulkerson and the Dinitz algorithm. Starting from a given flow, both algorithms, increase it until the maximum flow is reached. The Ford-Fulkerson algorithm achieves this by means of augmented paths, while Dinitz algorithm achieves this by means of auxiliary networks and "blocking flows".

As large share of combinatorial problems can be displayed and solved using a simplified version of networks with 0-1 flows, we will analyze the complexity of the Dinitz algorithm for that type of networks. Afterwards we will show and explain several application of network flows in solving selected combinatorial problems: Maximum matching in bipartite graphs and PERT digraph problems.

Finally, we use two practical problems of maximum matching in bipartite graphs: Worker distribution and Speed dating, to show how they can be solved using network flows. Here we will rely and test efficiency of two algorithms presented in previous chapters Dinitz algorithms and algorithms of Ford and Fulkerson.

Životopis

Juraj Šikonja rođen je 31. 7. 1990. u Zagrebu. Od mladih dana pokazuje interes za matematiku i nakon završene XV. ginmnazije u Zagrebu 2009. je upisao Preddiplomski studij Matematike na Prirodoslovno-matematičkom fakultetu Sveučilišta u Zagrebu. Preddiplomski studij koji završava 2013. godine. Nakon toga je upisao Diplomski studij Računarstva i matematike na istom fakultetu. Studij uspješno završava i ovim radom bi trebao diplomirati u 2017. godini.

Stručni studij ekonomije na Zagrebačkoj školi ekonomije i managementa upisao je paralelno sa preddiplomskim studiom Matematike 2010. godine kao vanredni student. U sklopu studija je jedan semestar proveo na studentskoj razmjeni na Queensland University of Technology, Astralija. Titulu stručnog prvostupnika ekonomije, bacc. oec. je stekao 2015. godine.

Nakon završenog studija Ekonomije zapošljava se 2015. godine u kompaniji McKinsey & Company u kojoj i danas radi kao Associate.

Juraj osim hrvatskog tečno govori engleski i njemački jezik.